Look-Ahead Simulation of Apparel Manufacturing

Final Report

A Short Term Research Development Task
Proposed Under DLA900-87-D-0017
Delivery Order #0003

R.P. Pargas
Associate Professor of Computer Science

Clemson Apparel Research
Clemson University
Clemson, South Carolina

Table of Contents

1. Background and Research Objectives

2. Results

3. Conclusions

4. Summary

5. Appendices

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave Blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	7 June 1997	Final 10/1/88 - 4/30/94

4. TITLE AND SUBTITLE
Look Ahead Simulation of Apparel Manufacturing

5. FUNDING NUMBERS
DLA900-87-D-0017
Del Order # 0003

6. AUTHORS
Dr. Roy P. Pargas
Dept. of Computer Science

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Clemson Apparel Research / Defense Personnel Supply Center
Clemson University / 2800 South 20th Street
500 Lebanon Road / Philadelphia, PA 19101-8419
Pendleton, SC 29670

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The approach taken was to build a simulator of apparel manufacturing tool and to measure the impact that such a tool would have on actual operations. The simulator runs on top of a real-time system which continually monitors the progress of operations on the shop-floor. The real-time system collects a wide variety of information as it tracks bundles of cloth making progress from workstation to workstation. The real-time systems monitors and reports on information such as operator efficiencies, start and stop times of operations on each bundle, the amount of work produced by each operator, which operator is assigned to each workstation, and the status of each bundle, cut and order.

14. SUBJECT TERMS

15. NUMBER OF PAGES
64

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
unclassified	unclassified	unclassified	unclassified

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-1
298-102

1. Background and Research Objectives

Between 1990 and 1993, the Defense Logistics Agency funded a project to study ways in which government contractors, primarily apparel manufacturers, could better manage and control their shop-floor operations. The primary objective of this research project was to explore the feasibility of using a simulation of manufacturing to assist management in the administration and control of operations on the shop floor. The question that the researchers were trying to answer was rather simple:

> *Does a tool that simulates and predicts short term performance of shop-floor operations help a supervisor (or higher level manager) perform his or her job?*

Details on the overall goals and design of this simulation software are given in a paper entitled *Production Scheduling Through Distributed Simulation* given in Appendix A.

The approach taken was to build such a tool, **a simulator of apparel manufacturing**, and to measure the impact that such a tool would have on actual operations. Figure 1 shows a diagram of the overall design of the simulator. The simulator runs on top of a real-time system which continually monitors the progress of operations on the shop-floor. The real-time system collects a wide variety of information as it tracks bundles of cloth making progress from workstation to workstation. The real-time system monitors and reports on such information as:

1. operator efficiencies,
2. start and stop times of operations on each bundle,
3. the amount of work produced by each operator,
4. which operator is assigned to each workstation, and
5. the status of each bundle, cut, and order.

The simulator periodically takes a snapshot of the files produced by the real-time system and moves time forward using the information. In this manner, the simulator predicts what the performance of the shop-floor will be in the near future. The output of the simulator is an immediate report on a number of metrics. These metrics give the user (the shop-floor supervisor) a clear graphical report on what to expect in the near-term, thus allowing the user ample time to anticipate and to avert potential problems down the road.

The steps taken in this research project were as follows:

1. Interview several plant managers concerning their production goals. The objective is to collect a set of performance metrics representing a wide class of objective functions for optimization.

2. Develop the simulation software.

3. Demonstrate the software to apparel manufacturers, specifically to shop-floor supervisors and others directly involved in production.

4. Collect feedback regarding the perceived potential of such a software package.

The remainder of this report details each of the steps above and summarizes the results obtained by this study. Conclusions are presented, and recommendations for possible future work are outlined.

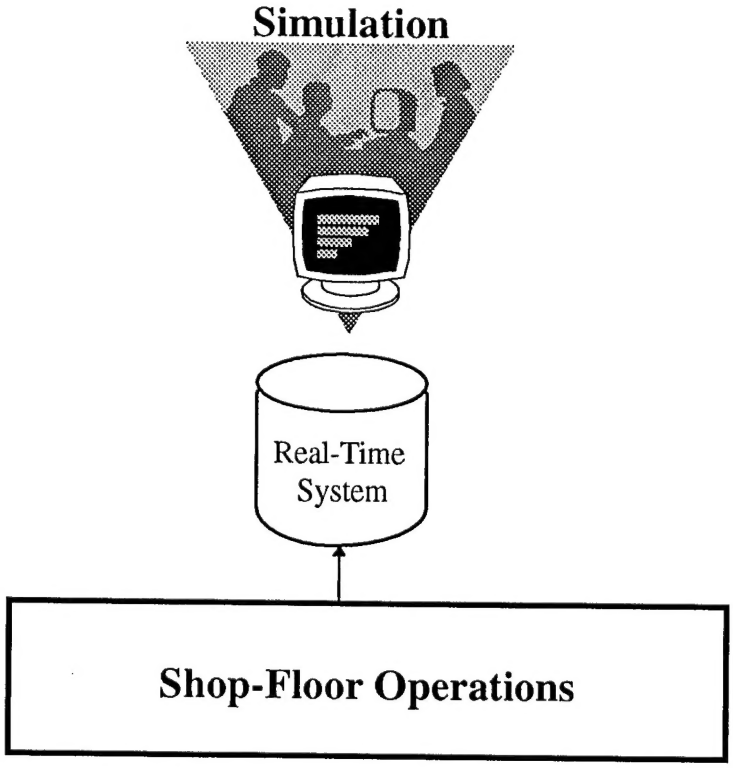


Figure 1

2. Results

2.1 Performance Metrics

The first data collected for this study were the performance metrics. These metrics were summarized by the researchers in a paper (see Appendix B) entitled *Shop-Floor Performance Metrics for the Apparel Industry* that was published in the *International Journal of Clothing Science and Technology 3,* No. 1, 1991.

A classification of the performance measures used in this study is given below:

1. *Waiting time.* Sub-assemblies often wait in queues before they are operated upon. A manager is interested in knowing whether a particular cut is being inordinately delayed or whether employees should be moved to places where excessive work has accumulated.

2. *Cost.* The expenditure incurred during job processing is identified by these metrics. As manufacturing proceeds, direct labor adds value to each product. When machines break down or employees wait idly for work or training is conducted, excess cost is accrued.

3. *Flow time.* This term quantifies the time jobs take to be worked on. One can compute the percentage of time that jobs wait, the average amount of value added to the jobs while they are on the shop-floor, etc.

4. *Lateness.* This gives the actual and estimated completion time and date of cuts and the amount by which they are early or late. Late jobs can incur penalties whereas jobs which are early may add to inventory costs.

5. *Machine Utilization.* The time wasted due to machine malfunction or machine idleness gives a measure of machine utilization.

6. *Labor Utilization.* The time spent by employees being productive or non-productive off-standard gives a measure of the utilization of labor. This time measure is similar to excess cost, a dollar measure.

7. *Production.* This measures work accomplished in terms of the SAMS (standard allowed minutes for an operation) produced.

8. *Efficiency.* This gives a comparative figure for actual output versus expected maximum output.

Each classification contains several metrics giving a total of seventy-nine metrics listed on page 20 of the paper in Appendix B.

A plant supervisor may select from among these seventy-nine metrics those which he or she feels is most important in his or her plant. All seventy-nine are calculated by the simulator by default.

2.2. Development of Simulation Software

The simulation software is complete. The software runs on a PC attached to a system of sixteen multiprocessors. It runs with a sample of data obtained from a real-time system and simulates the operations of a shop-floor for apparel manufacturing.

The output of the simulation software is a collection of seventy-nine metrics as described in Section 2.1 above. Any subset of the metrics may be selected by the user; the metrics are presented in graphical form.

The user may modify parameters of the software and may re-run the simulation. Modifications include adding or removing operators to the plant, adding or removing workstations, scheduling meetings, changing priorities of the cuts through the plant, scheduling new cuts for production, etc. Details on the various parameters available to the user are given in the paper entitled *Near-Term Distributed Simulation of Apparel Manufacturing* in Appendix C. These parameters allows the user to ask *what-if* questions and quickly (within one minute) view the results of the changes.

The software is mature and can be implemented at an apparel plant which already has a real-time system in place.

Some of the problems encountered in this project and associated with a distributed simulation, i.e., a simulation implemented on a multiprocessor system, are listed in papers entitled *Solving Synchronication Problems in Rapid Simulation of a Manufacturing Shop Floor* (Appendix D) and *Guidelines for Dynamic Load Balancing in Conservative Distributed Simulations* (Appendix E).

2.3. Demonstration of software to apparel manufacturers

The software was demonstrated to a large number of apparel manufacturing plant managers and administrators through seminars at Clemson Apparel Research and through visits conducted by the researchers at apparel plants in South Carolina and Alabama.

The administrators of one plant in particular, American Apparel, Inc. of Selma, Alabama expressed much interest in the simulation software. American Apparel has indicated that they would be very interested in having the simulation software implemented at plant as a *demonstration site*. If such an implementation were done, other government contractors would be able to view the simulation in use at an actual plant. This is discussed further in Sections 3.5 and 3.6 below.

3. Conclusions

There are six clear conclusions obtained from this study. They are listed and briefly described below.

3.1. *The design of the simulator is well-understood and mature.*

The design of the simulator evolved over the period of the research into a clean, clear, simple model. At the moment that the user wants to use the tool, the simulator makes a copy of all of the relevant files continually being updated by the real-time system. This is, in effect, a *snapshot* of the status of the shop-floor, including where each and every bundle is on the floor, how much work has been performed on it, how much work remains to be performed on it, which operator is currently assigned to each workstation, how efficient each operator is, whether an operator is on- or off-standard, and which workstations are in operation. These are the raw materials used by the simulator in performing its task.

The simulator simply takes all of this information and moves time forward, one unit at a time. At each time unit, the simulator applies each operator's efficiency to the bundle he or she is currently working on; this determines how much work is produced on the bundle in the time unit. As each bundle is completed, the simulator first decides where (to which operation) to send the bundle and then assigns a new bundle to the operator. Because the simulator knows the precise rate of work (efficiency) of each operator, it can accurately predict how much work flows through each workstation. This process is applied to each operator for a single time unit. The simulator then increments simulated time by one and repeats the process.

When simulated time reaches the final time specified by the user, the simulator will have the predicted the new status of each bundle on the floor, how much work was done on it during simulated time, how much work still remains, etc. In short, at the end of the simulated time interval, the files generated by the simulator comprise a snapshot of the shop floor at the future time specified by the user.

The process is simple and well understood. As a result, the results of the simulator are accurate and reproduceable.

3.2. *Inexpensive hardware technology available today has more than sufficient power to run the simulation..*

At the time of the research, a PC would have been too slow to provide the speed necessary for the simulation. As described in Section 3.1, each bundle must be moved forward at every time unit. This translates into hundreds of millions of computer operations for a shop-floor with 500 workstations and for a simulated time of five days. The research built the performance around a sixteen-processor multiprocessor system in order to provide the speed required. The decision was correct. The computation time to conduct a simulation with 500 workstation was consistently under one minute for a 5-day

simulation. On a single PC, the computation times for a similar simulation would range from 6 to 10 minutes.

Much has changed in PC technology since then. A single Intel Pentium Pro-based PC today outperforms the multiprocessor system used at the time of the research. If a real-time system is already in place at an apparel manufacturing plant, the only hardware requirement for the simulator is a Pentium Pro. The cost, therefore, is nominal; a $2500 system will cover all hardware requirements. This includes the 2 gigabyte hard disk that would be required to hold the significant number of files used by the simulator.

As shown in Figure 1, the simulator running on the PC is directly connected to the central server of the real-time system. When called upon to perform a simulation, the PC copies into its hard disk several files from the server. The simulator then begins operations on its own copies of the files, never disturbing the original real-time system files.

In the process of simulating shop-floor operations, the simulator changes the contents of the files. For example, in simulated time, bundles are moved from workstation to workstation. This must all be reflected in the bundle information file. Simply put, the simulator *takes a snapshot of the shop-floor and moves simulated time forward very rapidly.* At the end of the simulation, summary reports in the form of graphs of the metrics are presented to the user giving the user an accurate prediction of what the shop-floor looks like in the near future.

3.3. *The perceived and potential benefits of such a tool for an apparel manufacturer are substantial.*

"Control is the key to proper plant administration. And the right information provides that control." This was told to the researchers on this project by one of the plant managers with whom they spoke.

The simulator gives the plant supervisor this control. The supervisor does not need to predict what will happen in the long term, i.e., in two weeks or more. The supervisor has to know what will happen in the next few hours, the next few days, or the next week. With such a capability, the supervisor will be able to anticipate and ideally avoid bottlenecks in the manufacturing process.

On the shop-floor, such bottlenecks can appear for any of a number of reasons: (a) an employee at a critical operation may not show up for work that day, (b) a workstation may have broken down and will not be available for another 24 hours, (c) the new piece of equipment just installed may be producing piece goods at a much faster rate than the older equipment it replaced, (d) several new employees may still have low efficiencies due to lack of experience, etc. An experienced supervisor in a small plant will be able to anticipate and even prevent manufacturing problems due to some of these occurrences, but in a large (greater than five hundred workstations) manufacturing plant, it is unrealistic

to expect the supervisor to be able to know ahead of time all possible scenarios resulting from an unusual or unexpected event.

A simulation tool alleviates this situation significantly because the supervisor can play *"what if"* games, modifying the parameters of the simulation and evaluating their effects. The tool is powerful because it provides *the right information* to the shop-floor supervisor, and thereby offers *control.*

3.4. *The eagerness of apparel manufacturers for a simulation tool is great.*

Virtually every plant manager with whom the investigators spoke expressed interest in having such a simulator implemented at their plant. The managers with whom the investigators spoke most seriously were (1) Coastal Carolina, Inc. (2) National Apparel, Inc., and (3) American Apparel, Inc., all located in Selma, Alabama. Of the three, only American Apparel is currently operating at a high-enough level of production that could take advantage of a simulation tool.

Moreover, the investigators presented the concept of a simulation tool at many seminars at Clemson Apparel Research. The response from managers of the larger manufacturing plants was universal: simulation software would be a very powerful tool for anticipating and avoiding manufacturing problems.

Ramtex, Inc., a textile manufacturing plant in Ramseur, NC, has approached researchers at Clemson Apparel Research discussing the feasibility of implementing a similar simulation tool adapted, not to apparel, but to textile manufacturing. The administrators at Ramtex have a problem with scheduling because the demand for their textile and thread is so great that they are operating near 100% capacity. As a result it is critical that they schedule new orders very carefully. If they are not careful, they will not make full use of their equipment and therefore not be able to make their delivery deadlines. To them, the need for a simulation tool that assists in scheduling is of utmost importance.

3.5. *The manufacturing sophistication of at least one apparel manufacturer is at a level ready for simulation software.*

The situation at American Apparel, Inc., of Selma, Alabama is *ideal* for the implementation of a simulator. There are several reasons for this.

1. The plant is technologically sophisticated.

2. The plant already has a real-time system. There is a project funded by the Air Force currently being implemented at American Apparel. Part of the project involves implementation of a real-time system at the plant. Real-time systems are already in place both at the Selma and the Fort Deposit plants.

3. The plant administration is very willing to work with a simulator in production planning. Mr. J. Hodo of American Apparel has long expressed interest in the implementation of the simulator developed by Clemson Unversity at the Selma plant after the completion of the Air Force-sponsored project. He has verbally told researchers at Clemson University that American Apparel would be very willing to participate in a project that resulted in the implementation of a simulator for manufacturing.

4. The size of American Apparel is large enough to justify a simulator.

5. The company is a government contractor. Mr. J. Hodo of American Apparel has also said that he would be willing for American Apparel to become a *showcase* for other government contractors; officials from other companies could visit American Apparel to see how a real-time system and a simulator can be combined to give plant administration a tool for predicting near-term plant performance.

6. The timing is *perfect.* The Air Force project at American Apparel will be completed by the end of June 1997. The timing is perfect for the installation of a simulator to work with the real-time system already in place.

3.6. *The implementation of the simulation tool at American Apparel, Inc. can be performed at nominal cost.*

Because the technology is already developed, the software mature, and the implementation well-understood, the cost of implementing the simulation would be nominal. The time to implement and test could be done within six months. For example, if the implementation were started on 1 January 1998, the simulator would be completely installed, tested, and in daily use at American Apparel by 30 June 1998.

4. Summary

The simulation software proposed in this project is complete.

1. It runs on a PC attached to a system of multiprocessors.

At the time that the software was developed, multiprocessors were required to provide the speed that the simulation required. Because of the development of PCs since that time, the use of multiprocessors is no longer required. *The software can efficiently run on a standard Pentium-based PC running Windows NT or Windows 95.* The computation speed provided by current PCs is sufficient to provide the response time required by the simulation.

2. The simulation has been received well by managers and plant supervisors of apparel manufacturing plants.

Virtually all apparel plant managers and administrators, to whom the simulation software was demonstrated, appreciated the benefits that such a tool can provide. The software could be used immediately by plant supervisors in their day-to-day management of an apparel plant.

Plant managers of a textile manufacturing plant, Ramtex, Inc. of Ramseur, NC have also expressed interest in such a simulation tool and have discussed the possibility of implementing such software in their plant.

3. The software is mature and the design is well-understood.

It can be implemented immediately at an apparel plant which already has a real-time system implemented.

4. One candidate plant, American Apparel, Inc., of Selma, Alabama, is an ideal plant to implement the software.

American Apparel already has a real-time system implemented. The administrators of American Apparel have expressed great interest in participating in a project that would implement the simulation at their plant and to demonstrate the use of the simulation to other government contractors across the country.

Recommendation:

The investigators recommend that the simulation software developed in this project be implemented at American Apparel. Such an implementation should take no more than six months and can be performed at nominal cost.

Appendix A

Proceedings of the Fourth International Conference

Expert Systems in Production and Operations Management

Martin D. Goslar
Editor

May 14-16, 1990
Hilton Head Island, South Carolina

Sponsored by
Management Science Department
College of Business Administration
University of South Carolina

In cooperation with
American Association for Artificial Intelligence
Operations Management Association • TIMS College on Production and Operations Management
Daniel Management Center, College of Business Administration, University of South Carolina

PRODUCTION SCHEDULING THROUGH DISTRIBUTED SIMULATION

Roy P. Pargas and John C. Peck
Department of Computer Science
Clemson University
Clemson, SC 29634-1906

EXTENDED ABSTRACT

1.0 INTRODUCTION

Clemson University holds a contract with the Defense Logistics Agency to help improve the competitiveness of the US apparel industry through the use of advanced manufacturing technology and associated management techniques. As part of this contract, $1,500,000 each year is available for research projects intended to improve the productivity of apparel equipment and personnel. The authors of this paper, in cooperation with researchers at the University of Southwestern Louisiana, are engaged in one of several such research projects to develop a simulation software tool intended to assist middle to high level management in planning and scheduling work, personnel, and machines in the "near" term. The near term could be the next week, next day or next hour, depending on the frequency of use of the software tool.

1.0 THE PROBLEM

Apparel plants who have scheduling problems operate on a larger scale than manufacturing plants in many other industries. A typical apparel plant of this type would have more than 400 direct labor (incentive) employees with perhaps 500 machines (some large plants have 1500+ employees). Both employees and machines are capable of performing multiple operations, but only a small percentage of the total operations required to manufacture a particular garment. Active on the shop floor at any one time might be 100 or more production lots (orders) each consisting of perhaps 200 bundles of garment parts, each consisting of 5 or more subassemblies, each requiring 1 to 20 operations. Production lots are possibly of different styles, meaning the operations and sequencing of operations are different. Bundle subassemblies flow through the manufacturing process in parallel and join (merge), as operations are completed, to produce finished garments. The matching of subassemblies from parent bundles is important since color shading variations will be noticeable otherwise. Since employees and machines have multiple, but limited, skills and capabilities, load balancing of these resources against required work is a major problem.

2.0 PERFORMANCE METRICS

Since the ultimate goal in production scheduling is to improve the operation of a plant, the first question which must be asked is "How do you know if a change in an operational plan produces a better or worse schedule?" The performance goals of

management, and at different plants at different points in time, might produce drastically different answers to the question. At one point in time, the goal might to be maximize the output of size 16/34 red button-down collar shirts while at another time it might be to minimize the work-in-process inventory level subject to keeping the average efficiency of employees above 95%. In fact, the answer might change if management knew the optimization limits on the scheduling goals they might pick! The point of this discussion is that in general, one cannot always predefine an objective function for optimization. The process of measuring performance, while working toward optimizing any objective function, might change the function. This is a variation of what the physicists call the Heisenberg Uncertainty Principle or what the industrial engineers call the Hawthorne Principle. The process of measuring an entity changes the behavior of the entity being measured.

The first step taken in this research project was to interview numerous plant managers concerning goals to produce a set of performance metrics representing a wide class of objective functions for optimization. This set currently contains about 75 metrics which relate to the performance of the plant, each department, each lot and each style in-process. Employee performance employee pay, standard minutes produced, progress of specific work units, excess costs, lateness of orders and work-in-process inventory levels are but a few of the variables measured. This set of performance metrics is maintained during the execution of the simulator and displayed dynamically in the form of graphs as the values change. Multiple windows allow the simulator user to monitor any collection of metrics (subject to screen size limitations) as the simulation progresses.

3.0 SIMULATION AS A NEAR-TERM PLANNING TOOL

The approach taken by many simulations is to use statistical distributions to approximate variables such as job arrival and service time as well as employee efficiency, etc. The primary reason for this approach is that no real data is available to provide accurate values for these variables. The approach taken in this simulation is to use the large volume of data produced by a real-time shop floor control system (marketed by Foxfire Technologies Corporation and installed in several apparel plants in the US), which represents the current state of the apparel plant, to define the current state of the simulation. This real-time system collects data through use of a device at each workstation concerning employees, lots, bundles, subassemblies, and operation as work progresses through the plant.

The operation of the simulator is as follows. The simulation operator will define a plan for operation of the plant in the near-term (see above). The starting point for the plan is the current state of plant as represented by data contained in the real-time system. The operator then specifies changes to the plan which he wishes to implement at given times. New work might arrive at mid morning, personnel might be reassigned to different operations at specified times, machines might be taken out of service for preventive maintenance, new machines which operate at greater speeds might be placed into (simulated) service, etc. With this new plan for operation (usually a small variation in the

real plan currently in effect) the simulation will begin execution. The operator can request that the plant be simulated until a specified point in time, for a specified duration or until an interrupt key is pressed. During this period of time the graphical display of performance metrics will be dynamically updated. If the operator is unhappy with the values of certain metrics, he might choose to "rollback" the simulation to a specified point in time and rerun with a new plan. By this interactive and iterative technique, a plan which best suits management can be developed. This plan can then be printed and used as an order (or recommendation) to shop floor supervisory personnel. Monitoring of the actual shop floor performance is already provided as part of the Foxfire system.

The primary objective, therefore, is to allow the user to develop a production scheduling plan interactively, in the same manner as one works with a spreadsheet program. In a spreadsheet, the user can change a single cell, and every other cell whose value depends upon it changes automatically. The user can immediately see the "bottom line" result of his single change. The user of this simulation will be able to enjoy the same capability, i.e., to be able to change one part of his current production plan (say, replace Mary with Suzi on Machine A at 10:00 a.m.) and be able to see, very quickly, the effect of the change on overall production. The information the user receives will be the metric information he has selected to view. If he is happy with the new results, the user may opt to keep the employee change in his plan. Otherwise, he may delete the change and continue try other modifications to his plan.

4.0 DISTRIBUTED SIMULATION

A major concern in this simulation, however, is computer response time. A simulation of a large apparel manufacturing plant with 500 or more workstations is anticipated to require much more computing power than is available on a PC, even on the latest models such as the IBM PS/2 family, some of which use the very fast Intel 80386/387 processors. For this reason, the simulation is being developed on a distributed memory multiprocessor system. The multiprocessor consists of seventeen INMOS T-800 processors, called Transputers. Each Transputer has two Mbytes of memory and is slightly more powerful than an Intel 80386/387. Acting as a front-end processor which accepts input from, and provides graphics output to the user, is a standard PC, a COMPAQ Deskpro 386/25. The design calls for the simulation to execute entirely on sixteen of the seventeen Transputers. The results are collected by the seventeenth Transputer and sent to the PC for graphical display. Such a system should be able to deliver the speed necessary to make this simulation useful.

4.1 Synchronization Problems

The decision to develop a distributed simulation created two new problems which do not exist on single processor implementations. The first results from the fact that each processor is working asynchronously with its own logical clock. The processors communicate with each other by passing messages. Each of these messages carries a time-

stamp, the simulated time at which the information in the message was generated. If a processor gets too far ahead in simulated time, it is possible that it will receive a message from another processor in the (simulated) past. Had the processor received the message earlier, the simulation within this processor may have proceeded differently. The result, of course, is a potentially incorrect simulation.

Several solutions to the synchronization problem in distributed simulations have been proposed in the literature. One class of solutions, called conservative or pessimistic, allows each processor to proceed only when it is guaranteed that there is no possibility of receiving a message out of time-stamp order. One specific algorithm suggests that processors generate and send null, or empty, messages which carry nothing more than a time-stamp. The purpose is simply to broadcast each processor's current time to other processors. Although this does solve the problem, it also clogs the communication links with numerous null messages, eventually choking the system. Another class of solutions, called optimistic, suggests that processors operate as fast as they can without regard to synchronization. Should one processor receive a message out of time-stamp order, it initiates a process of "rolling-back" the simulation to a point where all processors are again in synch. Once in synch, processors again proceed independently. The drawback with this approach is, of course, the complications which arise when the program tries to undo a sequence of events and to restore the state at some earlier time. The approach implemented in this simulation is between the optimistic and the pessimistic solutions. Null messages are sent, but only on demand. If Processor A cannot proceed because it is unsure of the current simulated time of Processor B, it sends a request to B. Processor B will respond by sending a null message to A containing B's current time. With the new information, A can then proceed.

4.2 Load Balancing Among Processors

A second problem is that of load balancing. The ideal in a multiprocessor system is to balance the computational load evenly among all processors. This load, of course, is dynamic and cannot be predicted accurately. Effort must be made, therefore, as the simulation is running to monitor how much computation is actually done by each processor. When imbalance is detected, the simulation self-corrects, i.e., simulation workstations and buffers are sent from one processor to another thereby reducing the computation load of one processor and increasing the load of another.

5.0 CURRENT STATUS AND FUTURE PLANS

The simulation is, at present, approximately fifty percent complete. The project is anticipated to be completed by May, 1990 at which time test versions will be installed at two apparel plants, Tultex Corporation in Martinsville, Virginia and Jantzen Corporation in Seneca, South Carolina. Testing and modification should take approximately eight months, with a final version complete by December, 1990.

The industry currently has no simulation tool of this nature and thus has no means of performing near-term scheduling through the use of simulated production. Consequently, no expertise exists in the use of such a tool in near-term production planning. Since any expert system presumes the existence of an expert to help define the rules and methods for decision making, the second phase of this project will be to observe the way in which planners use the tool effectively and then produce an expert system which models their skills in the use of the simulation tool. The observations gathered by this means will result in the creation of an expert system to advise less skilled planners in making scheduling decisions. The final phase of the project may be to remove the user from the system (there is yet to be good evidence to suggest this is possible) and have the system create a plan with minimal human interaction during the process.

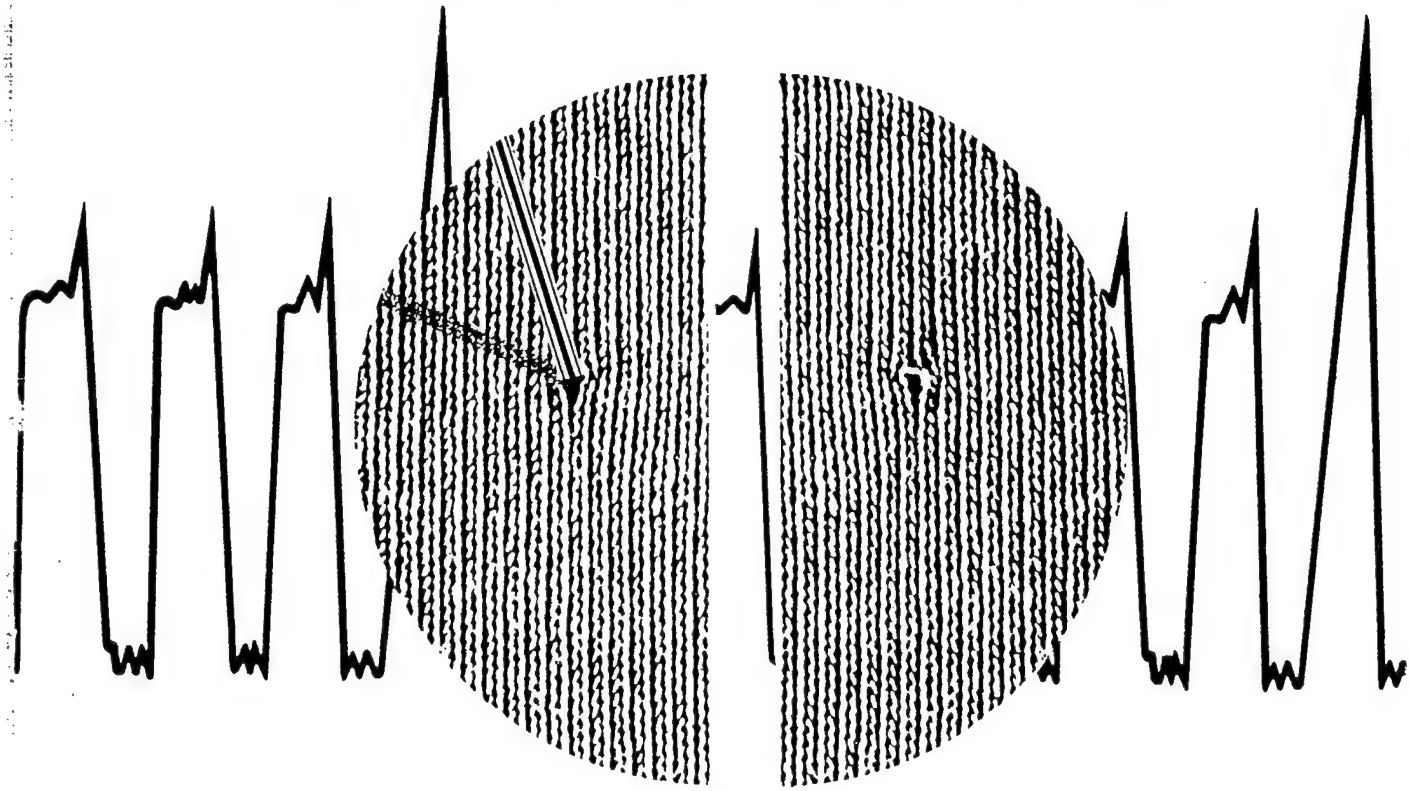
Appendix B

ISSN 0955-62

VOLUME 3 NUMBER 1 1991

INTERNATIONAL JOURNAL OF

CLOTHING SCIENCE AND TECHNOLOGY



MCB University Press

CONTENTS

VOLUME 3 NUMBER 1 1991

International Journal of Clothing Science and Technology

ABSTRACTS & KEYWORDS 2

FRENCH & GERMAN ABSTRACTS 3

EDITORIAL 4

OBJECTIVE MEASUREMENT OF FABRIC MECHANICAL PROPERTY AND QUALITY: ITS APPLICATION TO TEXTILE AND CLOTHING MANUFACTURING
S. Kawabata and Masako Niwa 7

SHOPFLOOR PERFORMANCE METRICS FOR THE APPAREL INDUSTRY
John C. Peck, Roy P. Pargas, Prashant K. Khambekar and Satish K. Dharmaraj 19

PERCEPTION OF TEXTURE, VISUALLY AND TACTUALLY: AN EXPLORATORY STUDY USING MULTIDIMENSIONAL SCALING ANALYSIS
Joan Laughlin 28

CENTRE SECTION

OBJECTIVE MEASUREMENT TECHNOLOGY BULLETIN FOR THE TEXTILE AND CLOTHING INDUSTRIES

VIEWPOINT i

PROGRESS STATUS REPORT i

COMMUNICATIONS ii

ANNOUNCEMENTS iv

submitted to International Journal of Clothing Science
and Technology

Shop-Floor Performance Metrics for the Apparel Industry

John C. Peck Roy P. Pargas Prashant K. Khambekar
Satish K. Dharmaraj
Department of Computer Science, Clemson University
Clemson, S.C. 29634-1906

May 8, 1990

Abstract

In a typical apparel plant, there may be hundreds of machines, hundreds of employees and thousands of bundle sub-assemblies all simultaneously active. It is the responsibility of the manager to optimize throughput and/or make efficient use of employee and machine resources and/or keep work-in-process inventory levels low. Ideally, to be able to deal with this complexity of scheduling, a manager would like to be presented with a handful of numbers which indicate how his/her plant is operating. The purpose of this paper is to identify and quantify a set of performance metrics which provide a manager with information concerning plant performance.

This paper motivates and presents a mathematical analysis of many factors which concern plant performance. The metrics enable a large amount of information to be condensed to summary form so that performance of an operational plan can be easily understood and lead to swift evaluation of that plan. The set contains about 70 metrics which the manager is free to interpret according to his/her operational goals.

Keywords: Apparel Industry, Production Management, Shop-floor Control, Scheduling, Performance Metrics, Simulation

Shop-Floor Performance Metrics for the Apparel Industry*

John C. Peck Roy P. Pargas Prashant K. Khambekar
Satish K. Dharmaraj
Department of Computer Science, Clemson University
Clemson, S.C. 29634-1906

May 7, 1990

INTRODUCTION

In a typical apparel plant, there may be hundreds of machines, hundreds of employees and thousands of bundle sub-assemblies of differing styles all simultaneously active. Work orders have costs and due dates associated with them and a shop-floor manager must assign employees to operations and set bundle priorities, that is, develop a day-to-day *operational plan* to complete the work orders. It is the responsibility of the manager to optimize throughput while making efficient use of employee and machine resources. Alternatively, he/she may strive to keep work-in-process inventory levels low. However, because of the complexity of scheduling, the manager is faced with a welter of confusing information, and, not too infrequently, crisis situations. Ideally, a manager would like to be presented with a single number or a handful of numbers which indicate how his/her plant is currently operating or how the operation of his/her plant would change as plans change. The purpose of this paper is to identify and quantify a set of performance metrics which provide a

*This work was supported in part by DOD Contract No. DLA900-87-D-0017 Task No. 0003

manager with information concerning plant performance.

These metrics enable a large amount of data to be compacted and easily understood. They are styled after scheduling metrics as in French [1] and Rinnooy Kan [2] and were produced after interviews with numerous plant managers at various levels of responsibility concerning goals and a wide class of objective functions for optimization. The set contains about 70 metrics which the manager is free to interpret according to his/her operational goals. The complete set of metrics are presented along with the motivation and rationale for incorporating each.

The paper is organized in the following manner. The definitions of commonly used terms are followed by the types of metrics. Before all the metrics are motivated and derived, some basic quantities which are not metrics in themselves but are used to define metrics are given. A summary is given at the end.

GENERAL DEFINITIONS

The sewing room has four distinct types of entities: garment parts, styles, employees and machines.

A *cut* is a lot or production order consisting of garment parts and has a due date. Usually, many cuts of the same *style* are being processed simultaneously in the plant. Styles are defined by style flow graphs which indicate the operations, the expected time for performing each of the operations and the sequence in which operations must be performed on garments. A cut is composed of many *bundles* of garment parts which can be in varying stages of completion. A bundle usually consists of components for (typically) 12 to 100 garments and may be divided into different *sub-assemblies* (such as a collar sub-assembly and a sleeve sub-assembly for a shirt) being worked on simultaneously. (Unit Production System clamps can be considered as bundles of size 1.)

The employees of a plant are partitioned into *departments*. Usually work in one department is independent of another.

A *job* refers to a unit of work and hence, in this document can mean a cut, a bundle or a sub-assembly as indicated by a subscript. Each sewing operation has an estimated standard time for its completion and is measured in Standard Allowed Minutes (SAMS). SAMS for an operation are determined by industrial engineers through time studies.

Employees are usually paid on an incentive scheme. They are said to be *on-standard* while working normally on a sub-assembly. During abnormal periods, such as machine failure, they are said to be *off-standard*. Employees differ in the skills they possess and the efficiency of performing operations.

The words shop-floor and plant are used interchangeably in this document.

TYPES OF METRICS

A manager is interested in whether work will be finished in time for delivery, how machines and employees are being utilized, whether any department in the plant is lagging behind and how the plant is doing overall. Thus performance metrics can be separated into four levels: *Cut level*, *Style level*, *Departmental level* and *Plant level*. Unless otherwise specified, each metric is defined for all four levels.

The metrics enable the manager to see whether work was uniformly produced during the day or whether there were any slack periods or periods swamped with work. The metrics can be calculated for any particular time period of interest (e.g. from 8 a.m. to 11 a.m.). Such a time period is simply called *the period*.

A classification of the performance measures is given below.

1. Waiting time. Sub-assemblies often wait in queues before they are operated upon. A manager is interested in knowing whether a particular cut is being

inordinately delayed or whether employees should be moved to places where excessive work has accumulated.

2. Cost. The expenditure incurred during job processing is identified by these metrics. As manufacturing proceeds, direct labor adds value to each product. When machines break down or employees wait idly for work or training is conducted, excess cost is accrued.

3. Flow time. This term quantifies the time jobs take to be worked on. One can compute the percentage of time that jobs waited, the average amount of value added to the jobs while they are on the shop-floor, etc.

4. Lateness. This gives the actual and estimated completion date of cuts and the amount by which they are early or late. Late jobs can incur penalties whereas jobs which are early may add to inventory costs.

5. Machine Utilization. The time wasted due to machine malfunction or machine idleness gives a measure of the machine utilization.

6. Labor Utilization. The time spent by employees being productive or non-productive off-standard gives a measure of the utilization of labor. This time measure is similar to excess cost which is a dollar measure.

7. Production. This measures work accomplished in terms of the SAMS produced.

8. Efficiency. This gives a comparative figure for actual output versus expected maximum output.

All metrics are motivated and described here and the manager must decide which metrics to give more importance depending on his/her priorities. How to interpret the metrics is up to the manager.

Before the metrics are given, symbols used in the document are presented.

SYMBOLS

Since the number of symbols is high this section tabulates the symbols used in the document. First, the basic entities or subscripts:

(symbol)	(units)	(description)
b	-	index into bundles of a cut
c	-	index into cuts of a style
d	-	index into departments of a plant
e	-	index into employees
$final$	-	the last operation of a job
	-	for incomplete jobs, it is carried out in a future period
fjp	-	first job of the period for a machine
fop	-	first operation of the period for a job
i	-	index into idle times of an employee
j	-	index into jobs in the order seen by a machine
ljp	-	last job of the period for a machine
lop	-	last operation of the period for a job
m	-	index into machines
o	-	index into operations
s	-	index into sub-assemblies of a bundle
z	-	index into styles in a plant

Now some auxiliary symbols:

(symbol)	(units)	(description)
dur	min	duration of period in minutes
n	-	the number of jobs
tdy	date	today's date
w	-	weight (either priority weight or value weight)

Now the metric related symbols:

(symbol)	(units)	(description)
BD	date	begin date of jobs
CD	date	completion date of completed jobs
CLI	min	clock-in time of an operation w.r.t. machine
CLO	min	clock-out time of an operation w.r.t. machine
DD	date	due date of jobs
DMS	$-min	dollar-min sewn for a job from its start
DMW	$-min	dollar-min waiting for a job from its start

DV	\$	dollar value at the end of the period
DVO	\$	dollar value at the end of each operation
DVP	\$	dollar value for the period
E	days	earliness of jobs
EC	\$	excess cost of whole plant
ECD	date	estimated completion date of incomplete cuts
EFF	SAMS/min	efficiency of an employee
EFM	SAMS/min	maximum efficiency of an employee
EL	days	estimated lateness of incomplete jobs
FP	\$	formula pay
FTP	min	flow time for the period
IT	min	idle time of machines
L	days	lateness of jobs
MFT	min	mean flow time of jobs
ML	days	mean lateness of jobs
MPP	SAMS	maximum possible production
MWT	min	mean waiting time of jobs
NOP	min	non-productive off-standard time of employees for the period
NR	\$/SAMS	on-standard rate (dollar per SAM) of operation
NWC	-	normalized waiting cost
NWT	-	normalized waiting time
OBR	\$/SAMS	operation base rate (for each operation of a job)
OR	\$/min	off-standard rate of employee
OT	min	off-standard time of employees
POP	min	productive off-standard time of employees for the period
PP	\$	production pay
PPI	-	plant productivity index
PSC	-	percent SAMS completed
$RWTFT$	-	ratio of waiting time to flow time
$RSTFT$	-	ratio of sewing time to flow time
$RSVFT$	-	ratio of SAMS value to flow time
$RSVST$	-	ratio of SAMS value to sewing time
SC	\$	standard cost of jobs
SMS	SAMS-min	SAMS-min sewn
SMW	SAMS-min	SAMS-min waiting
ST	min	sewing time of jobs if subscripted then for that operation
STP	min	sewing time of jobs during the period
SV	SAMS	SAMS value from start of job till end of the period
SVH	SAMS	SAMS value produced per hour
SVO	SAMS	SAMS value from start of job till end of an operation
SVP	SAMS	SAMS value from start of the period till end of period
T	days	tardiness of jobs
TDM	\$-min	total dollar-min for a job from its start

TSM	SAMS-min	total SAMS-min
TSV	SAMS	total SAMS at the end of the job
WFT	min	weighted sum of flow times
WIT	min	weighted sum of idle times
WL	days	weighted sum of latenesses
WT	min	waiting time of jobs before an operation if subscripted then for that operation
WTP	min	waiting time of jobs during the period
WWT	min	weighted sum of waiting times

Symbols may be subscripted with b, c, d, e, i, j, m, o, s or z as defined above. A symbol with a subscript represents the quantity for that particular entity e.g. SV_c indicates the SAMS Value of cut c whereas SV_b indicates SAMS Value of bundle b (of some cut). A Symbol without any subscript represents the quantity for the whole plant.

Note: The value of any quantity with a subscript of 0 is 0.

CALCULATIONS OF SAMS VALUE

Some quantities are basic and referenced repeatedly and are thus discussed before the discussion on metrics. SAMS Value calculations are given in this section whereas Dollar Value calculations are given in the following section.

Each operation has attached to it a SAMS value which is a measure of work (time) expected from a standard employee (i.e. one with 100% efficiency).

SAMS Value at the end of an operation (SVO)

The measure of accumulated work from the start of a job till the end of a particular operation o is

$$SVO_{s,o} = SVO_{s,o-1} + SAMS_o \tag{1}$$

where $SAMS_o$ is the SAMS for operation o and is a known quantity.

SAMS Value (SV)

SV measures the accumulated work from the start of a job till *the last operation of the period* (which is represented by *lop*). SAMS value for sub-assembly s, SV_s, is obtained by summing over all the operations in that sub-assembly.

$$SV_s = \sum_{o=1}^{lop} SAMS_o \tag{2}$$

SV_b is obtained by summing over all the sub-assemblies in that bundle.

$$SV_b = \sum_{s} SV_s \tag{3}$$

SV_c is obtained by summing over all the bundles in that cut.

$$SV_c = \sum_{b} SV_b \tag{4}$$

SV_z is obtained by summing over all the cuts in that style.

$$SV_z = \sum_{c} SV_c \tag{5}$$

SV (for the plant) is obtained by summing over all the styles in the plant.

$$SV = \sum_{z} SV_z \tag{6}$$

Note: Since other metrics are similarly sums of their components, they will not be repeatedly derived. A summary list of metrics appears at the end.

SAMS Value Produced in the Period (SVP)

SVP is the quantity produced only during the period of interest. This indicates how much work was accomplished and can be used to compare with a target or to calculate efficiency.

$$SVP_s = \sum_{o=1op}^{lop} SAMS_o \tag{7}$$

$$SVP_b = \sum_{s} SVP_s \tag{8}$$

Total SAMS Value (TSV)

TSV is the total work put into a job when it is completed. Since all the operations are known even before work starts on the job, this is a known quantity. This quantity is a goal to be achieved.

$$TSV_s = \sum_{o=1}^{final} SAMS_o \tag{9}$$

$$TSV_b = \sum_{s} TSV_s \tag{10}$$

CALCULATIONS OF DOLLAR VALUE

As opposed to the SAMS value, which measures time invested (or to be invested) in a job, Dollar Value measures the investment or cost.

Dollar Value at the end of an operation (DVO)

The measure of accumulated worth from the start of a job till the end of a particular operation o is

$$DVO_{s,o} = DVO_{s,o-1} + SAMS_o * NR_o \tag{11}$$

Dollar Value (DV)

DV measures the accumulated worth from the start of a job till *the last operation of the period.*

$$DV_s = \sum_{o=1}^{lop} (SAMS_o * NR_o) \tag{12}$$

$$DV_b = \sum_{s} DV_s \tag{13}$$

Dollar Value Produced in the Period (DVP)

DVP is the dollar worth produced only during the period of interest.

$$DVP_s = \sum_{o=fop}^{lop} (SAMS_o * NR_o) \tag{14}$$

$$DVP_b = \sum_{s} DVP_s \tag{15}$$

WAITING TIME

The Waiting Time is the time jobs must wait in the buffers before being processed. A large value indicates bottleneck problems in the plant i.e. a pile up of work in certain areas on the floor suggesting that some employees may be moved there to alleviate the load.

Waiting Time for the period (WTP)

WTP is the total waiting time during the period of interest.

$$WTP_s = \sum_{o=fop}^{lop} WT_o \tag{16}$$

$$WTP_b = \sum_{s} WTP_s \tag{17}$$

Normalized Waiting Time (NWT)

Some jobs are almost complete whereas some jobs have just begun. The jobs which are almost complete will generally have more money invested in them so a manager may want to weight the waiting time in proportion to the invested standard minutes.

SAMS-min Waiting, SMW, is defined first and is the waiting times for operations weighted with the SAMS investment in the job prior to those operations.

$$SMW_s = \sum_{o=fop}^{lop} (SVO_{s,o-1} * WT_o) \tag{18}$$

$$SMW_b = \sum_s SMW_s \tag{19}$$

Next SAMS-min Sewn, SMS, is defined as the sewing times for operations weighted with the SAMS investment in the job prior to those operations.

$$SMS_s = \sum_{o=fop}^{lop} (SVO_{s,o-1} * ST_o) \tag{20}$$

$$SMS_b = \sum_s SMS_s \tag{21}$$

So Total SAMS-min, TSM, is the total flow time of the operations weighted with the SAMS investment in the job prior to those operations.

$$TSM_c = SMW_c + SMS_c \tag{22}$$

A normalized waiting factor weighted according to SAMS value is thus obtained.

$$\mathrm{NWT_C} = SMW_c/TSM_c \tag{23}$$

COST

Costs are of importance to everyone. Standard cost, excess cost and the total value of work-in-process inventory are the different types of costs.

Standard Cost (SC)

Standard Cost, SC, is the amount of money paid out during the period to employees for working on jobs (as against for being off-standard). SC is identically equivalent to DVP, the expression of which is derived in the preliminary calculations of Dollar Value.

Excess Cost (EC)

Excess Cost, EC, is the cost due to machine breakdowns, employees unavailable due to meetings to attend, etc. which causes employees to be placed in off-standard

statuses. This is applicable at the plant level only.

$$PP_e = \sum_o SAMS_o * OBR_o \tag{24}$$

$$FP_e = \sum_o (\%clocktime * hrs * OR + \%SAMS * SAMS_o * OBR_o) \tag{25}$$

$$\text{EC} = \sum_e (\max(PP_e, FP_e) - PP_e) \tag{26}$$

In-process Inventory Cost

In-process Inventory Cost indicates the amount of worth that is tied up in incomplete jobs. It is identically equal to DV, the expression of which is derived in the preliminary calculations of Dollar Value.

Normalized Waiting Cost (NWC)

Similar to weighting the waiting time according to the standard minutes invested in a job, a manager may wish to weight the waiting time according to the amount of money invested in the jobs. The jobs which are almost complete will generally have more money invested and will get weighted more.

Dollar-min Waiting, DMW, is defined first and is the waiting times for operations weighted with the dollar investment in a job prior to those operations.

$$DMW_s = \sum_{o=fop}^{lop} (DVO_{s,o-1} * WT_o) \tag{27}$$

$$DMW_b = \sum_s DMW_s \tag{28}$$

Dollar-min Sewn, DMS, is defined next as the sewing times for operations weighted with the dollar investment in the job prior to those operations.

$$DMS_s = \sum_{o=fop}^{lop} (DVO_{s,o-1} * ST_o) \tag{29}$$

where, ST_o, the sewing time for the operation is,

$$ST_o = SAMS_o / EFF_{o,e} \tag{30}$$

Now,

$$DMS_b = \sum_s DMS_s \tag{31}$$

So Total Dollar-min, TDM, is the total flow time of the operations weighted with the dollar investment in the job prior to those operations.

$$TDM_c = DMW_c + DMS_c \tag{32}$$

A normalized waiting factor is thus obtained (but one which is weighted according to dollar value).

$$\mathrm{NWC_C} = DMW_c / TDM_c \tag{33}$$

FLOW TIME

Flow Time of a job is the total amount of real time spent in the plant. Since jobs are either being operated upon or waiting, the Flow Time is the sum of these two times.

Flow Time for the period (FTP)

FTP, the flow time for the period, is the sum of sewing time for the period and the waiting time for the period.

$$FTP_s = STP_s + WTP_s \tag{34}$$

where,

$$STP_s = \sum_{o=1op}^{lop} ST_o \tag{35}$$

Now,

$$FTP_b = \sum_s FTP_s \tag{36}$$

Note that for an incomplete sub-assemblies which begins at the start of the period, FTP is identically equal to *dur*, the duration of the period. FTP will be smaller than *dur* for sub-assemblies which get completed during the period or incomplete sub-assemblies which start at some time during the period.

There are, additionally, four simple ratios of interest:

$$RWTFT_c = WTP_c/FTP_c \tag{37}$$

RWTFT ranges from 0 to 1 as the waiting time is a part of the flow time.

$$RSTFT_c = STP_c/FTP_c \tag{38}$$

Again, RSTFT ranges from 0 to 1 as the sewing time is a part of the flow time.

$$RSVFT_c = SVP_c/FTP_c \tag{39}$$

RSVFT ranges from 0 to ∞ as employees can have greater than 100% efficiency, theoretically infinite efficiency producing infinite SAMS.

$$RSVST_c = SVP_c/STP_c \tag{40}$$

Again, RSVST ranges from 0 to ∞.

LATENESS

The manager wants to quantify lateness, as excessive lateness may have penalties. This measures how good the plant is at being able to finish cuts by their due dates. For cuts already completed, one has a concrete Completion Date; for incomplete cuts, the Completion Date must be estimated.

Completion Date (CD)

For cuts which are completed during the period,

$$CD_c = tdy \tag{41}$$

Estimated Completion Date (ECD)

For cuts which are still incomplete at the end of the period, the completion date is an educated guess. Assuming that the rate of work on the cut will be the same, a simple proportion of work completed to total work to be done gives the Completion Date.

$$\mathrm{ECD_C} = BD_c + TSV_c * (tdy - BD_c)/SV_c \tag{42}$$

Lateness (L)

For cuts which are completed during the period,

$$\mathrm{L_C} = CD_c - DD_c \tag{43}$$

$$\mathrm{L_Z} = \sum_c L_c \tag{44}$$

Estimated Lateness (EL)

For cuts which are still incomplete at the end of the period, the corresponding rough measure is,

$$\mathrm{EL_C} = ECD_c - DD_c \tag{45}$$

$$\mathrm{EL_Z} = \sum_c EL_c \tag{46}$$

Earliness and Tardiness (E and T)

Lateness can be either positive or negative. Lateness which is purely negative is termed Earliness whereas Lateness which is purely positive is termed Tardiness.

Hence,

$$\mathrm{E_C} = \begin{cases} L_c & \text{if } L_c < 0 \\ 0 & \text{otherwise} \end{cases} \tag{47}$$

$$\mathrm{E_Z} = \sum_c E_c \tag{48}$$

$$\mathrm{T_C} = \begin{cases} L_c & \text{if } L_c > 0 \\ 0 & \text{otherwise} \end{cases} \tag{49}$$

$$\mathrm{T_Z} = \sum_c T_c \tag{50}$$

MACHINE UTILIZATION

Idle Time (IT)

A machine is idle between the end of an operation and the beginning of a new one. A machine is also idle in the event of a breakdown. Idle Time conversely gives a measure of utilization. ($CLO_{fjp-1} = CLO_0$ = start time of the period.)

$$IT_m = \sum_{j=fjp}^{ljp} (CLI_j - CLO_{j-1}) \tag{51}$$

$$\mathrm{IT} = \sum_m IT_m \tag{52}$$

LABOR UTILIZATION

Employees can be on productive off-standard or non-productive off-standard. These time measures conversely give the labor utilization.

Productive Off-std Time for the period (POP)

POP is the time due to machines working imperfectly, employees working on samples, employees being trained, etc., which causes employees to go off-standard. This

is applicable for the department and plant level only. (Here OT is *productive* part of off-standard time.)

$$\mathrm{POP_d} = \sum_e \sum_i OT_{e,i} \tag{53}$$

$$\mathrm{POP} = \sum_d POP_d \tag{54}$$

Non-productive Off-std Time for the period (NOP)

NOP is the time due to machine breakdown, employee unavailable due to meetings, etc., which causes employees to go off-standard. This is applicable for the department and plant level only. (Here OT is *non-productive* part of off-standard time.)

$$\mathrm{NOP_d} = \sum_e \sum_i OT_{e,i} \tag{55}$$

$$\mathrm{NOP} = \sum_d NOP_d \tag{56}$$

PRODUCTION

Production is a pure SAMS measure as opposed to Standard Cost which is a dollar measure.

SAMS Value Produced in the Period (SVP)

The amount of production is the SAMS value produced in the period, SVP, the expression of which is derived in the preliminary calculations of SAMS Value.

SAMS Value Produced per Hour (SVH)

$$\mathbf{SVH_c} = SVP_c/(dur/60) \tag{57}$$

Percent of SAMS Completed (PSC)

PSC measures the amount of work done till the end of the period and may be compared with a target for production.

$$\mathrm{PSC_C} = SV_c * 100/TSV_c \tag{58}$$

EFFICIENCY

Plant Productivity Index (PPI)

PPI is a rough measure which indicates how well the employees are performing assigned operations as compared to the best historical performance on any operation. An employee has a maximum efficiency in a particular operation and the Maximum Possible Production, MPP, is achieved if all employees are assigned to the operation they do best. As the name indicates this is applicable only at the plant level.

$$EFM_e = \max_o EFF_{e,o} \tag{59}$$

$$MPP = dur * \sum_e EFM_e \tag{60}$$

$$\mathrm{PPI} = SVP/MPP \tag{61}$$

MEANS

Since the number of jobs a plant processes varies from time to time, the following averages are of interest.

$$\mathrm{MWT} = WTP/n \tag{62}$$

$$\mathrm{MFT} = FTP/n \tag{63}$$

$$\mathrm{ML} = (L + EL)/n \tag{64}$$

WEIGHTED SUMS

Some jobs may have higher priorities or more value than other jobs. In general, there may be a weight attached to each quantity. Hence, weighted quantities are of interest. These weights can vary from 0 to 1.

$$\text{WWT} = \sum_c WTP_c * w_c \tag{65}$$

$$\text{WFT} = \sum_c WFT_c * w_c \tag{66}$$

$$\text{WL} = \sum_c L_c * w_c \tag{67}$$

$$\text{WIT} = \sum_m IT_m * w_m \tag{68}$$

SUMMARY

The question of how a plant manager determines if his/her plant is performing well still cannot be answered in general; however, this paper has motivated and presented a mathematical analysis of many factors which concern plant performance. A summary list of metrics is given in Table I below.

As can be seen the performance metrics enable a large amount of information to be condensed to summary form. Performance of an operational plan can be easily understood and lead to swift evaluation of that plan.

One of the research projects being conducted at Clemson University is the development of a near-term simulation of apparel plant shop-floor operation to aid a plant manager in creating and evaluating plans [3]. Given an operational plan and the current state of the shop-floor the activities during the course of a day are simulated in a few minutes. The performance metrics presented in this document are computed on the fly and graphically displayed. These metrics are thus an integral part of a very practical tool.

	Cut Level	Style Level	Dept. Level	Plant Level
Waiting Time	WTP_c NWT_c	WTP_z NWT_z	WTP_d	WTP NWT
Cost	DVP_c DV_c NWC_c	DVP_z DV_z NWC_z	DVP_d DV_d	DVP EC DV NWC
Flow Time	FTP_c $RWTFT_c$ $RSTFT_c$ $RSVFT_c$ $RSVST_c$	FTP_z $RWTFT_z$ $RSTFT_z$ $RSVFT_z$ $RSVST_z$	FTP_d $RWTFT_d$ $RSTFT_d$ $RSVFT_d$ $RSVST_d$	FTP RWTFT RSTFT RSVFT RSVST
Lateness	L_c EL_c E_c T_c	L_z EL_z E_z T_z		L EL E T
Machine Utiliz				IT
Labor Utiliz			POP_d NOP_d	POP NOP
Production	SV_c SVP_c SVH_c PSC_c	SV_z SVP_z SVH_z PSC_z	SVP_d SVH_d	SV SVP SVH PSC
Efficiency			PPI_d	PPI
Means				MWT MFT ML
Weighted Sums				WWT WFT WL WIT

Table I. Summary List of Metrics.

REFERENCES

1. French, S., *Sequencing and Sheduling: An Introduction to the Mathematics of the Job-Shop*, Ellis Norwood, London, 1982.

2. Rinnooy Kan, A. H. G., *Machine Scheduling Problems: Classification, Complexity and Computations*, Martinus Nijhoff, The Hague, 1976.

3. Pargas, R. P., Peck, J. C., Khambekar, P. K., Dharmaraj, S. K., "Near-term Distributed Simulation of Apparel Manufacturing", submitted to the *Western Simulation Multiconference*.

Appendix C

1990
WINTER SIMULATION CONFERENCE
PROCEEDINGS

Edited by:

OSMAN BALCI
Virginia Polytechnic Institute and State University

RANDALL P. SADOWSKI
Systems Modeling Corporation

RICHARD E. NANCE
Virginia Polytechnic Institute and State University

9 – 12 December 1990
The Fairmont Hotel
New Orleans, Louisiana

Proceedings of the 1990 Winter Simulation Conference
Osman Balci, Randall P. Sadowski, Richard E. Nance (eds.)

NEAR-TERM DISTRIBUTED SIMULATION OF APPAREL MANUFACTURING

Roy P. Pargas
John C. Peck
Prashant K. Khambekar
Satish K. Dharmaraj

Department of Computer Science
Clemson University
Clemson, South Carolina 29634-1906

ABSTRACT

This paper describes the design and implementation of a near-term simulator for apparel manufacturing. The primary purpose of the simulator is to provide management with a tool for production planning. In order to run as accurate a simulation as possible, data describing the status of the apparel plant is continually collected by a real-time shop-floor control system. This data is then used as input to the simulator. In order to increase the speed of the simulator and reduce response time, the simulator is implemented on a multiprocessor system consisting of seventeen T-800 INMOS Transputers with a COMPAQ Deskpro 386/25 serving as front-end processor. A user enters a plan for the next production period and receives back more than seventy different measures of performance of shop-floor operations. The user may iterate this process, making modifications to the plan, until the simulated shop-floor performance is satisfactory. The final result is a detailed plan for the next production period.

1. INTRODUCTION

Apparel plants with scheduling problems operate on a larger scale than manufacturing plants in many other industries. A typical apparel plant of this type may have more than 400 direct labor (incentive) employees with perhaps 500 machines (some large plants have 1500+ employees and a correspondingly larger number of machines). Both employees and machines are capable of performing multiple operations, but only a small percentage of the total operations are required to manufacture a particular garment. Active on the shop-floor at any one time might be 100 or more production lots (orders) each consisting of perhaps 200 bundles of garment parts, each bundle consisting of five or more subassemblies, each subassembly requiring one to twenty operations. Production lots are possibly of different styles, meaning the operations and sequencing of operations are different. Bundle subassemblies flow through the manufacturing process in parallel and join (merge), as operations are completed, to produce finished garments. The correct matching of subassemblies from parent bundles is important since color shading variations will be noticeable otherwise. Since employees and machines have multiple, but limited, skills and capabilities, load balancing of these resources against required work is a major problem.

The ultimate goal in production scheduling is to improve the operation of a plant. The primary question is "How does one know if a change in an operational plan produces a better or worse schedule?" The performance goals of management, at different plants at different times, may produce very different answers to the question. At one point in time, the goal may to be maximize the output of size 16/34 red button-down collar shirts, whereas at another time, it may be to minimize the work-in-process inventory level subject to keeping the average efficiency of employees above 95%. With different goals, different objective functions, and different management styles, the question then is: how does one build a single management tool to facilitate production scheduling?

This paper describes one attempt at building such a tool: *a near-term simulator of shop-floor operations of an apparel manufacturing plant.* We set four objectives for the simulator:

(1) It should accurately predict near-term performance ("near-term" is loosely defined as one hour to five days).

(2) It should be possible to use the simulator interactively and iteratively. The user should be able to change values of the simulation and observe the effects quickly. Very short response time is therefore necessary.

(3) The simulator should have wide applicability, i.e., it should be usable in different plants with different performance goals.

(4) The simulator should be easy to use, and the results easy to read and comprehend.

The simulator, which is conservative and event-driven [Bryant 1977; Chandy and Misra 1979; Peacock et al. 1979] is described in detail in the rest of this paper. Section 2 shows how one may use a simulator as a tool for production planning. Section 3 describes the user interface, both input and output. Section 4 describes a set of performance metrics available to the user. Section 5 gives an overview of the design and implementation of the simulation. Section 6 discusses conclusions and future plans.

2. SIMULATION AS A PLANNING TOOL

Most simulations are beset with one major problem: the difficulty of obtaining actual data on which the simulation can operate. As a result, simulation designers attempt to estimate crucial information such as the rate of arrival of goods to be processed, processing rates of different machines, and skill levels (efficiencies) of employees. The results predicted by the simulation will, of course, be close to reality only if the estimates are accurate. However, coming up with good estimates can often be both difficult and frustrating.

An alternative approach is to measure, in advance, the processes normally estimated by many simulations. For example, in an apparel manufacturing plant, one may measure the skill level of each operator on each different machine type. Or one need not estimate the rate of arrival of goods into the plant if one knows exactly what orders are arriving and at what time. In apparel manufacturing, as in many industries, skill levels vary widely from employee to employee. Assuming that all employees perform at some average rate or according to some statistical distribution may seriously handicap the ability of a simulator to make accurate predictions. One can also take note that, for example, on a given day, a one-hour company-wide meeting will be called to discuss employee benefits; the effect, of course, is that production stops completely for that one hour. If it is known that a high-priority order of goods is to be started in the morning, one may mark the lot associated with the order accordingly, causing the lot to be sewn ahead of others already on the shop-floor. In short, every piece of information that can be measured *is* measured. Estimates of missing data are made only if there is no way to measure the information directly.

This on-going measurement requires that a real-time shop-floor control system be in place. Such a real-time system can collect data through a network of intelligent devices, one at each workstation on the shop-floor. These devices measure and record

a variety of facts, a few of which are:

(1) which bundles have arrived at which operation (this provides accurate tracking of every bundle currently in process),

(2) the number of minutes it takes for Employee A on Workstation 1 performing operation P on Bundle X (this provides one data point which will contribute to a measure of the skill level of this particular employee at this operation, as well as keeps track of the amount of work this employee has done),

(3) the number of bundles on the floor (providing a global view of the number and distribution, across the shop-floor, of bundles of parts),

(4) the number of minutes a particular workstation has been idle (giving up-to-the-minute information on utilization of equipment).

The data are transmitted from the workstations, through the network, and are collected on a PC for viewing or for processing (for example, to produce a payroll at the end of each day or week).

The real-time system employed in this study is one developed and marketed by Foxfire Technologies Corporation [Foxfire Technologies Corporation 1989]. This system, currently installed in a number of apparel manufacturing plants across the country, essentially provides detailed information on every employee, every lot, and every workstation on the shop-floor.

3. THE USER INTERFACE

The simulator, a block diagram of which is shown in Figure 1, operates as follows. The user defines a plan for operation of the plant in the near-term, i.e., for the next one or several days. The starting point for the plan is the current state of the plant as represented by data provided by the real-time system. The user then specifies changes to the plan which he or she wishes to implement at given times. New work may arrive at midmorning, personnel may be reassigned to different operations at specified times, machines may be taken out of service for preventive maintenance, new machines which operate at greater speeds may be placed into (simulated) service, etc. With this new plan for operation (usually a small variation of the real plan currently in effect) the simulation begins execution. The operator can request that the plant be simulated until a specified point in time, for a specified duration or until an interrupt key is pressed.

Figure 2 shows the instructions available to the user when developing an input plan. The parameters associated with each are not shown, in order to reduce clutter. The selected instructions are entered by the user and saved in a file which serves as input to the simulator. The instructions have been grouped by function. In the first group, Initialization, there are instructions to start and end the simulation at specified (simulated) times. Instruction 3 names the files produced by the real-time system. These files provide information on the current status of the plant. Instructions 4 and 5 inform the simulator of changes in personnel. For instruction 5, it is necessary to specify which workstation the now-present employee will be assigned to; this is because the simulator must look up this employee's efficiency on the specified workstation in the database provided by the real-time system.

The next group of instructions inform the simulator of events that will occur during the day. For example, (6) a new lot will arrive at a certain time, (7) a meeting is scheduled later in the day, or (8) a regular plant-wide break will take place. Instructions 9 and 10 schedule events relating to specific employees; an employee is paid at a different rate as a result of employee training, or an employee is moved from one workstation to another because of anticipated bottlenecks in the workflow. Instructions 11 and 12 allow changes made to a workstation. Instruction 13 modifies the lot priority, allowing it to move faster or slower through the plant.

Instructions 14 through 17 provide the user with control over the simulator itself: pausing, restarting, and stopping. Instructions 18 through 28 give a variety of information on employees, workstations, and pay code tables. Buffers refer to the queues of bundles of parts waiting to be sewn. Finally, instructions 29 through 31 are warning signals for which the user may request. The user may want to be alerted when the simulator signals that an employee has no more work, that an entire buffer (which typically provides bundles of parts to several workstations) is empty, or that an empty buffer has just received a new set of bundles.

During the simulation, a set of approximately seventy-five performance metrics (Section 4) are maintained. These metrics measure different aspects of plant operations displayed as column graphs. The user may select up to four of these for graphical display on the computer monitor (Figure 3). The user has available the typical graphics window capabilities: opening and closing of windows, bringing windows to the front, line scrolling up and down, page scrolling, moving windows around the screen, and resizing. One window is used for input and enables the user to enter a plan, whereas the rest are output windows showing a variety of information (see items 18 through 28 of Figure 2). User input menus prompt the user for specific information whenever needed. Finally, a help window gives information on all other windows. All window graphics routines are implemented with Borland's Turbo-C graphics library calls.

As the simulation progresses, the display graphics are dynamically updated. If the simulation operator is dissatisfied with the values of certain metrics, he or she may choose to restart the simulation after making changes to the current plan. In this interactive and iterative manner, a plan which best suits management can be developed. This plan can then be printed and used as a recommendation to shop-floor supervisory personnel. We believe that in the beginning a user will experiment with different sets

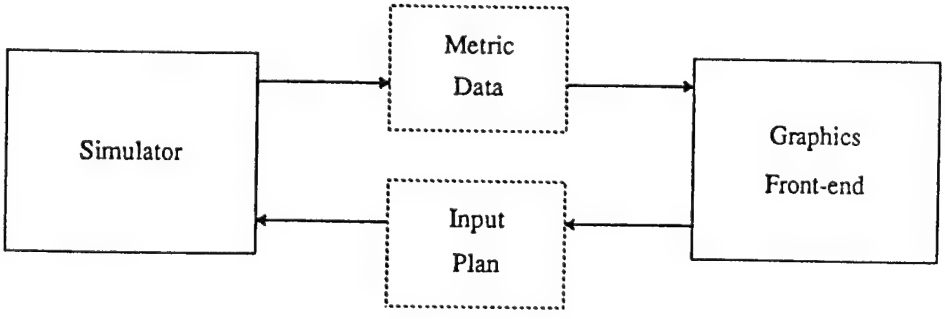


Figure 1. Block Diagram of Simulator

INITIALIZATION
1. Start time
2. End time
3. Input data files (RTS files)
4. Employee present during last period (yesterday) but absent now
5. Employee absent during last period (yesterday) but present now

SHOP-FLOOR MANAGEMENT
6. New lot arrival
7. Schedule meeting
8. Set time of break
9. Change pay code
10. Move employee to another workstation
11. Add an operation to a workstation
12. Delete an operation of a workstation
13. Change priority of a particular lot

SIMULATOR CONTROL
14. Stop immediately
15. Resume simulation
16. Stop at a given time
17. Restart

LISTS
18. Show list of off-standard emplyees
19. Show list of on-standard emplyees
20. Show list of idle employees
21. Show list and status of all employees
22. Show list of skills and efficiencies of an employee
23. Show list of workstations attached to a buffer
24. Show list of empty buffers
25. Show list of idle workstations
26. Show list of assigned workstations
27. Show list of machines and operations of a workstation
28. Show Pay Code Table

INTERRUPTS
29. Inform user that an employee is idle
30. Inform user that a buffer is empty
31. Inform user that work has arrived at a previously empty buffer

Figure 2. Commands Available to the User

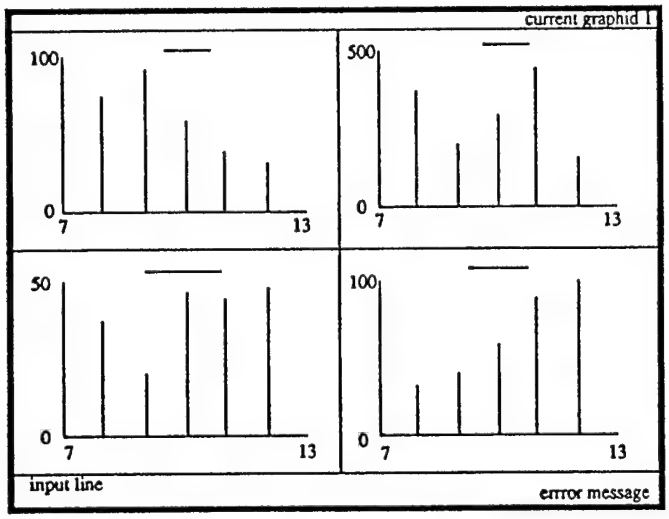


Figure 3. The Normal Graphics User Interface

of metrics, but that over time, he or she will settle on one set which will be used thereafter.

A useful time to use the simulator is at the end of a shift, or just before a shift starts. During this period, the user has time to work with the simulator and plan the operations for the next shift. However, because a snapshot of the current status of the shop-floor is available at any time (a benefit of using a real-time shop-floor control system), it is possible for a user to run the simulation at virtually any time of the day using as the current snapshot of the plant that information most recently provided by the real-time system. The simulation gives the user a tool for quickly predicting the performance of the shop-floor during the remainder of the current production period.

In summary, this simulator provides the user with a tool for developing a production scheduling plan interactively. The manner one uses this tool is very similar to the manner one uses a spreadsheet program. With a spreadsheet, the user can change a single cell, and every other cell whose value depends upon it changes automatically. The user can immediately see the "bottom line" result of a single change. Similarly, the user of this simulation can change one part of the current production plan (say, replace one employee with another on Workstation 1 at 10:00 a.m.) and be able to see, very quickly, the effect of the change on overall production for some future period of time. The feedback the user receives is the performance metric information selected for viewing. If he or she is satisfied with the new results, the user

may opt to keep the employee change in the plan. Otherwise, the user may delete the change and may continue trying other modifications.

4. PERFORMANCE METRICS

One design decision was to provide a wide variety of performance metrics for the user to select and view. The reason for this is the belief that different managers will want to optimize different metrics. Some may want to maximize workstation utilization. Others may try to minimize amount of work-in-process inventory. Still others prefer minimizing employee idle time. As a result, the best approach is to provide all of the metrics and let the user select those which he or she wishes to observe and improve. We expect that after some experience with the simulation has been gained, the user will settle on a subset of the metrics thereafter. As an aside, note that some metrics conflict and thus cannot be optimized simultaneously. For example, in order to keep employees working on production, it may be necessary to increase the amount of work-in-process.

The metrics used in this study were derived largely from concepts developed for job-shop scheduling, as described by French [1982] and Rinnooy Kan [1976]. Metrics are provided at four different levels: lot (a single production order), style, department, and plant. Seven classes of metrics are available to the user:

(1) Waiting time. Bundles of parts generally wait in queues before they are sewn. A manager may be interested in knowing whether a particular lot (composed of several bundles) is being inordinately delayed or whether employees should be moved to operations where excessive work has accumulated.

(2) Cost/Value. As a lot moves through the plant, it accumulates value. Each sewing operation adds labor value. Machine breakdown, lack of work, employee training, and the like, add excess cost to plant operation but add no value to the product.

(3) Flow time. This measures the amount of time a lot is on the shop-floor. One may compute the percentage of the time jobs are waiting in queues, or the average amount of dollar-value per minute each lot accrues while on the shop-floor.

(4) Lateness. Being able to estimate by how many days a lot will be late (or early) allows a user to plan ahead. For example, he may decide to give a higher priority to a lot to allow it to move along through the plant. On the other hand, a lot that will be completed too early may require unwanted inventory handling. The user may decide to hold processing on an early lot for a day or two.

(5) Labor utilization. Significant labor underutilization indicates several possible problems: workload imbalance, suggesting that the idle employees be trained to perform other operations; too little work-in- process, suggesting that more new orders be started; or simply too many employees for the current work available, suggesting that fewer employees be scheduled for the next several days.

(6) Production. This provides the user with a variety of measures for the amount of work produced and the rate at which it is produced.

(7) Efficiency. This provides a comparison of the actual output produced by the plant and the maximum possible output of the plant.

For a detailed description of the complete set of metrics used, the interested reader is referred to a paper by Peck et al. [1990].

5. DISTRIBUTED SIMULATION

A major concern in this simulation is simulation response time. A simulation of a large apparel manufacturing plant with 500 or more workstations is anticipated to require much more computing power than is available on a PC, even on the latest models such as the IBM PS/2 family, some of which use the very fast Intel 80386/387 processors. For this reason, we decided to implement the simulation on a distributed memory multiprocessor system.

The multiprocessor system used was built by Computer System Architects of Provo, Utah. It consists of seventeen INMOS T-800 Transputers, each with two megabytes of memory. The processor integrates a 32-bit processor, a 64-bit floating point unit, and 4 Kbytes of static RAM. At peak speeds, each processor provides 10 MIPS and 1.5 Mflops. Each has four communication ports which can be used to interconnect with other processors. Sixteen processors, therefore, can be linked to form a linear array, a ring, a mesh, a 2-D torus, or a 4-D hypercube. One of the Transputers is called the root processor and serves as a liaison between the front-end PC and the sixteen other Transputers, called nodes. The front-end processor, which accepts input from and provides graphics output to the user, is a standard PC, a COMPAQ Deskpro 386/25, with a 110 Mbyte hard disk.

A major design decision was to break up the major functions of the simulation among the processors available. The primary functions are input and output, execution of the simulation itself, sending of user input to the simulation, and collection and processing of performance metric data (Figure 1). A natural assignment of function to processor is to assign all input/output function to the front-end PC, the simulation to the Transputer nodes, and to let the root Transputer serve as a liaison, collecting metric data and broadcasting user input to the nodes.

The responsibilities of the front-end computer are to provide the interface allowing the user to enter and modify plans, and to display all metric information in graphical format. All shop-floor status information is stored in a database which resides on the PC hard disk. The root Transputer collects and processes performance metric data before sending the data to the front-end for display. As each node steps through the simulation, it accumulates performance metric information and, at predefined time intervals, sends the data to the root Transputer. The root collects the data, performs a few simple computations (such as computation of means and variances), and when the data for the time interval is complete, sends a packet of metric information to the front-end PC for graphic display.

This division by function is quite clean. The user sees only the front-end computer and does not know about the existence of the root or node Transputers. The program executing on the front-end is unaware of the number of processors running the simulation. It is only aware of simulator commands and data it sends down the communication link to the root Transputer, and metric and other information received from the root. In the same way, the root program is independent of the number of Transputer nodes and the nodes depend on the root only for startup information. The simulation may run on one node Transputer, or as many as the user can afford. For smaller problems, one Transputer may suffice. For larger problems with 500 or more workstations, the user may decide whether the increased speed of execution justifies the cost of additional processors. The design is flexible enough, however, to accommodate any number of node Transputers.

The Transputer nodes execute an event-driven simulation. Because physical memory is limited on each node Transputer (two megabytes per processor) and virtual memory is not provided by the node operating system, we have opted for a conservative simulation, rather than the optimistic Time Warp approach proposed by Jefferson [1985]. The most common event is completion of a sewing operation on a subassembly. Other events include lot events (arrival of a lot, change in a lot's priority), employee events (employee is assigned to another workstation, employee stops work to attend a meeting), and workstation events (a workstation is reconfigured for a different sewing operation).

Each sewing operation has a unique buffer which holds a queue of subassemblies waiting to be sewn. One or more workstations, configured to perform the operation, pick subassemblies from the buffer in a first-come, first-served order. A processor, one of the node Transputers, simulates the activity of one or more buffers. For correct first-come, first-served simulation, the workstations must coordinate. This is most easily achieved by re-

quiring that all workstations that pick from a common buffer be simulated in a single processor. Each processor has a single process, a single event queue, and a single logical clock. The event queue is stored in ascending order and, as a result all events within (and resultant messages from) a processor are in logical time sequence. Each node processor goes through the event queue, generating new events for itself and other processors, until the end of the simulation period.

Subassemblies may flow from a buffer in one processor to a buffer in a different processor. Buffers in a processor which can receive subassemblies from another processor are called Front Buffers. For correct conservative simulation, a processor cannot proceed if there is a possibility of receiving a subassembly message from a predecessor processor at an earlier clock time. This means that if correct time synchronized simulation is to be guaranteed, all Front Buffers of a processor must be non-empty (other buffers may be empty) before a processor advances its simulation clock. One way to handle this is for predecessor processors to send NULL messages to successor processors. However, source-driven NULL messages are likely to flood the system, as reported by Fujimoto [1988]. An alternative approach is: if the Front Buffer of a processor becomes empty, the processor sends out appointment-request messages to its predecessors. The predecessors then make an estimate based on their current statuses and send appointments to the requesting processor, thereby enabling it to proceed. This demand-driven method, described by Khambekar and Dharmaraj [1990], is an adaptation of the appointment approach presented by Nicol and Reynolds [1984] and Nicol [1988]. In terms of the design space outlined by Reynolds [1982], this method is accurate, non-aggressive, has no risk, and employs knowledge acquisition and knowledge embedding.

Subassemblies have user-assigned priorities and are arranged in the buffer queues according to their priorities and arrival times. Occasionally, two or more subassemblies must merge into one. For example, fronts and backs must merge to form a complete shirt. Hence, the mere arrival of a subassembly in a buffer does not guarantee that it is ready for processing; it may have to wait for its companion subassembly to arrive at the same buffer. Information on the flow of companion subassemblies is extracted and stored in compressed form prior to the start of the simulation. A simulation node ready to select a subassembly must scan the buffer queue, skipping over all unmatched companion subassemblies.

The service time for an operation in a subassembly is calculated from the pre-engineered Standard Allowed Minutes (SAMS) and the efficiencies of the employees. For example, if the SAMS value of an operation is 5 minutes and an employee has an efficiency of 110% (better than average), then the service time for this employee working on this operation is $5/1.1 = 4.55$ minutes. The completion time of the simulated operation is computed and an event is added to the event queue.

Subassemblies usually flow from one operation to another; however, sometimes there are alternate paths available for the subassembly. For example, instead of performing two consecutive operations on two slow older machines, it may be possible to combine both operations on one fast, newer machine. When alternate paths are available, a subassembly is sent along the path with the smallest wait-queue. This implies that the path of the subassembly must be determined at runtime, i.e., determining the successor buffer (which buffer to send the subassembly to next) is determined on-the-fly by examining the wait-queues of all the successor buffers. If the selected successor buffer happens to be in the same processor, then the subassembly is sent directly to the buffer; otherwise, additional table look-up is necessary to determine the successor processor, and a subassembly message is sent to that processor.

When an operation completion event occurs, the processor calculates all contributions to the collection of metrics. Results are stored in local memory. After a predetermined interval of simulated time, the processor sends the accumulated metrics to the root Transputer. The root accumulates these in its own global set of metrics. When all processors have sent metric values for a single time interval, the root computes a set of secondary metrics (means, variances, plant-wide totals) and sends the entire list of metric information to the host PC for immediate display.

6. SUMMARY

This paper describes a near-term simulator for use in scheduling production in an apparel manufacturing plant.

In the Introduction, we set four objectives for the simulator: accuracy of simulation, interactive and iterative use, wide applicability and ease of use. We feel these goals are being met. This simulation is accurate because little information is estimated. Current and historical data obtained from a real-time control system are used, rather than statistical distributions to estimate such things as rate of arrival of goods and employee skill level. Interactive and iterative use is due, in large part, to the division of tasks among multiple processors. The very short response time which results encourages the user to sit before the monitor and work with the simulator. Wide applicability is achieved because of the large number of performance metrics made available to the user. The user selects as many of the metrics as he or she feels are important in planning shop-floor operations. Finally, ease of use is a result of the graphics windowing system available; the user views many performance metrics changing over time.

We believe that this simulation will be a valuable tool for a manager of shop-floor operations.

ACKNOWLEDGEMENT

This research was sponsored in part by the Defense Logistics Agency under Contract No. DLA900-87-D-0017 Task No. 0003 through the Clemson Apparel Research Center.

REFERENCES

Bryant, R.E. (1977), "Simulation of Packet Communication Architecture Systems," Technical Report MIT/LCS/TR-188, MIT, Cambridge, MA.

Chandy, K.M. and J. Misra (1979), "Distributed Simulation: A Case Study in Design and Verification of Distributed Programs." *IEEE Transactions on Software Engineering SE-5*, 5, 440-452.

Foxfire Technologies Corporation (1989), *Real-time Shop-Floor Control System User Manual*, Marietta, GA.

French, S. (1982), *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*, Ellis Norwood, London.

Fujimoto, R.M. (1988), "Performance Measurements of Distributed Simulation Strategies," In *Proceedings of the 1988 SCS Multiconference on Distributed Simulation*, San Diego, CA, 14-20.

Jefferson, D.R. (1985). "Virtual Time," *ACM Transactions on Programming Languages and Systems 7*, 3. 404-425.

Khambekar, P.K. and S.K. Dharmaraj (1990). "Approaches to Solving Synchronization Problems in Parallel Simulation of an Apparel Plant," In *Proceedings of the 1990 ACM Southeast Regional Conference*, C.M. Pancake and R.M. Geist, Eds. ACM, Greenville, SC, 274-281.

Nicol, D.M. and P.F. Reynolds (1984), "Problem Oriented Protocol Design," In *Proceedings of the 1984 Winter Simulation Conference*, S. Sheppard, U. Pooch, and D. Pegden, Eds. IEEE, Dallas, TX, 471-476.

Nicol, D.M. (1988), "Parallel Discrete-Event Simulation of FCFS Stochastic Queuing Networks," In *Proceedings of the 1988 ACM SIGPLAN PPEALS*. Yale University, New Haven, CT, 124-137.

Peacock, J.K., J. W. Wong, E.G. Manning (1979), "Distributed Simulation Using a Network of Processors." *Computer Networks 3*, 1, 44-56.

Peck, J.C., R.P. Pargas, P.K. Khambekar, and S.K. Dharmaraj (1990), "Shop-Floor Performance Metrics for the Apparel Industry." Submitted to the *International Journal of Clothing Science and Technology*.

Reynolds, P.F. (1982), "A Shared Resource Algorithm for Distributed Simulation." In *Proceedings of the Ninth Annual International Computer Architecture Conference*, Austin, Texas, 259-266.

Rinnooy Kan, A.H. (1976), *Machine Scheduling Problems: Classification, Complexity and Computations*, Martinus Nijhoff, The Hague.

Appendix D

Artificial Intelligence and Simulation

Proceedings of the Simulation MultiConference on
Artificial Intelligence and Simulation
1-5 April 1991
New Orleans, Louisiana

Edited by
Ranjeet J. Uttamsingh
Synetics

A. Martin Wildberger
General Physics

Simulation Series
Volume 23
Number 4

Sponsored by
The Society for Computer Simulation (SCS)

SOLVING SYNCHRONIZATION PROBLEMS IN RAPID SIMULATION OF A MANUFACTURING SHOP-FLOOR

John C. Peck, Roy P. Pargas, Prashant K. Khambekar, Satish K. Dharmaraj
Department of Computer Science
Clemson University
Clemson, South Carolina 29634-1906

ABSTRACT

This paper describes a solution to synchronization problems which arise in rapid near-term sir ılation of manufacturing. The purpose of the simulator is to provide management with a tool for use in creating and evaluating schedules as part of production planning. In order to run as accurate a simulation as possible, data describing the current status of the manufacturing plant is continually collected by a real-time shop-floor control system. This data is used to start the simulator in an initial state. Performance of the simulation with quick run-time strongly suggest a parallel implementation; T-800 INMOS transputers are used with a PC as a front-end processor. Unlike a single processor simulation, a parallel simulation gives rise to synchronization problems, deadlock and starvation. These problems are analyzed and solutions which enable an accurate, conservative simulation are presented.

INTRODUCTION

Apparel plants operate on a larger scale than manufacturing plants in many other industries. A typical apparel plant may have more than 400 direct labor (incentive) employees with perhaps 500 machines (some large plants have 1500+ employees and a correspondingly larger number of machines). Both employees and machines are capable of performing multiple operations, but only a small percentage of the total operations are required to manufacture a particular garment. Active on the shop-floor at any one time might be 100 or more production lots (orders) each consisting of perhaps 200 bundles of garment parts, each bundle consisting of five or more subassemblies, each subassembly requiring one to twenty operations. Production lots are possibly of different styles, meaning the operations and sequencing of operations are different. Bundle subassemblies flow through the manufacturing process in parallel and join (merge), as operations are completed, to produce finished garments. The correct matching of subassemblies from parent bundles is important since color shading variations will be noticeable otherwise. Production lots have due dates by which they must be shipped out. Since employees and machines have multiple, but limited, skills and capabilities, balancing these resources against required work, that is, developing an operational plan, is a major problem.

The ultimate goal in production scheduling is to improve the operation of a plant. The primary question is "How does one know if a change in an operational plan produces a better or worse schedule?" A near-term simulator of shop-floor operations of an apparel manufacturing plans has been developed to enable the manager to evaluate such a plan. High-level performance information in the form of graphics is continuously displayed. The manager can evaluate the plan and in case the performance metrics are not satisfactory, rerun the simulation with a new plan, all in a matter of minutes. Details of the simulation can be found in Pargas *et al.* 1990.

Rather than using statistical distributions to estimate crucial input information (which many simulations do) the near-term simulator uses a real-time system (Foxfire 1989) to *measure*, in advance, information such as the rate of arrival of goods to be processed, processing rates of different machines and efficiencies of employees. Thus the question of accuracy of estimates does not arise.

Since production goals, such as keeping inventory low or machine utilization high, vary from plant to plant, or indeed, from time to time in the same plant, and management styles vary from person to person, a large number of performance metrics are made available. A manager may select and observe any of the metric graphs on the front-end and optimize performance based on them. The sixty or so metrics fall into different classes: Cost, Production, Efficiency, Lateness, Labor utilization and Waiting Time (Peck *et al.* 1991).

The complexity of the application, large input data, large number of metrics and quick simulation require-

ment necessitate a parallel simulation. The implementation is done on a multiprocessor system built by Computer Systems Architects of Provo, Utah. It consists of 17 INMOS T-800 Transputers, each with 2 Mbytes of memory. The Transputers integrate a 32-bit processor, a 64-bit floating point unit and 4 Kbytes of static RAM. One of the Transputers (called the root) is connected to a standard PC front-end whereas the other 17 Transputers are networked among themselves and connected to the root.

A major design decision was to break up the major functions of the simulation among the processors available. The primary functions are input and output, execution of the simulation itself and collection and processing of performance metric data. A natural assignment of functions to processors was to assign all input/output functions (that is, inputting the data and displaying graphs) to the front-end PC, the simulation to the Transputer nodes, and the collection and processing of metric data to the root Transputer. The design is clean: the user is unaware of the Transputers, the program on the PC is independent of the number of Transputers and the simulation program on the Transputers is independent of the front-end. The Transputer nodes execute a distributed event-driven simulation and exploit the natural parallelism on the shop-floor. Distributed simulation gives rise to synchronization problems, deadlock and starvation. The solution to these problems is presented in this paper.

THE APPAREL PLANT SHOP-FLOOR

Bundles of garment parts are sewn according to their respective style (operation) flow graphs. Figure 1 shows two style flow graphs. The circles represent operations to be performed, for example, operation 17 is "hem placket" and operation 41 is "top-stitch pocket". On the shop-floor each operation has a buffer and one or more workstations are configured to perform that operation. When employees are assigned to a workstation, a bundle is extracted from the workstation's buffer, the operation is performed and then the bundle is sent to the buffer corresponding to the next operation in the style flow graph.

Bundles carry with them the expected processing time for each operation in the style graph. This time, called Standard Allowed Minutes (SAMS), is obtained from engineering time-motion studies. Employees differ in their efficiency of performing operations. Efficiency is defined as the ratio of SAMS to actual minutes and can be above or below 100%. Hence, although the SAMS value of an operation is known in advance, the actual service time depends on which employee picks up the bundle, and is only available when the processing of the bundle is simulated.

There are three types of events: *bundle events* such as the completion of an operation or the arrival of a production lot, *employee events* such as an employee signing in for work or an employee being reassigned to another workstation, and *workstation events* such as a workstation being reconfigured for a different type operation. The most common event is the completion of an operation.

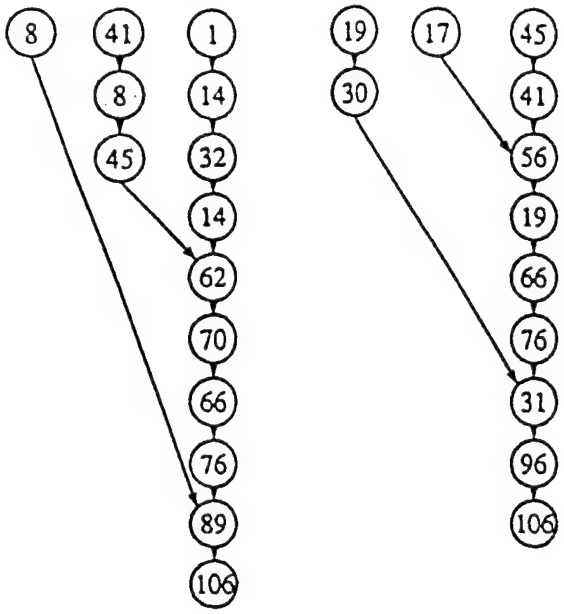
PARALLEL SIMULATION

For the simulation, operations are mapped to processors in the transputer system. A processor may simulate one or more operations. Each operation's buffer is represented by a queue and bundles are picked from the queue in order of priority and in case of equal priority, in order of time of arrival.

For correct simulation workstations configured for the same operation (and so drawing from the same buffer) must coordinate. This is most easily achieved by requiring that all workstations configured for the same operation be simulated in the same processor. Activities of all employees assigned to such workstations are also simulated in the same processor. Thus each processor has its independent queue of events and performs event-driven simulation synchronized by a logical clock. The logical clock is defined as the time of the last event processed by the processor.

A majority of researchers in distributed simulation use a logical process for each real process (e.g., workstation) in the application. Each logical process communicates with other logical processes. However as has been shown by Nicol (Nicol 1988), if the number of logical processes is greater than the number of physical processors, an overhead of context switching is experienced and efficiency is reduced. Hence in this simulation buffers, workstations and employees in one processor are handled by a single process. Since the event queue is stored in ascending time order all events within (and resultant messages from) a processor are in logical time sequence.

Transputer processors communicate with each other using messages. All messages are time-stamped, that is, they carry the logical clock time at which the sending processor issued the message. The majority of interprocessor messages are bundle messages which indicate the transfer of a bundle from one processor to a buffer in another processor for further processing according to the bundle's style flow graph. The sending processor is called the predecessor and the receiving processor



Style 1 Style 2

Figure 1: Typical Style Flow Graphs

is called the successor.

Although each style flow graph is feed-forward and has no cycles, the intersection of styles gives rise to cycles. As an example, in style 1 of Figure 1 operation 41 precedes operation 8 which precedes operation 45. In style 2 operation 45 precedes operation 41. Thus 41, 8 and 45 form a cycle of operations in the merged style graph. If such cycles were small, that is, composed of less than 8 operations (as compared to the total number of operations being in the range of 100), then operations which form cycles could all be mapped to the same processor, thus giving rise to a cycle-free interconnection among *processors*. Indeed this was the initial expectation (Khambekar and Dharmaraj 1990). However, examining the style flow graphs reveals that this is not feasible. Figure 2 shows the result of intersecting 21 style flow graphs. All operations in the big oval form a single cycle; thus 33 out of the 42 operations are involved in a cycle. Hence, cycles between processors cannot be avoided unless 75% or more of the simulation is performed in a single processor. As a means of distributing the workload, operations are mapped to processors such that the processors have equal number of operations.

SYNCHRONIZATION PROBLEMS

Since each processor has an independent event queue and logical clock, it is possible that the logical clock of a predecessor is greater than, equal to or less than the logical clock of the successor. If the logical clock of the predecessor is less than that of the successor, a message sent by the predecessor will arrive in the successor's simulated *past* and either compromise the accuracy of the simulation or incur additional overhead to undo earlier events. Accordingly, there are two approaches: *Optimistic* approaches (Jefferson 1985; Sokol *et al.* 1988) allow messages to arrive in the simulated past and in case such a message arrives, a roll-back of the simulation is initiated back to the time in the message. *Conservative* approaches prevent events from executing out of time sequence. Optimistic approaches involve keeping the state of the simulation at every step and sending anti-messages to cancel the effect of earlier non-chronological actions. Since keeping the states is memory-intensive and the amount of memory in each transputer is limited (compared to the complexity of the application) and virtual memory is not available, a conservative simulation approach was selected. Therefore, before a processor can select the next event, it must know that none of its predecessor processors will send a message with a lesser time-stamp. This restriction can cause the processor to wait for messages till the message time-stamps are equal to or greater than the time of the next event.

Two synchronization problems can occur with this conservative approach: *deadlock* and *starvation (no-*

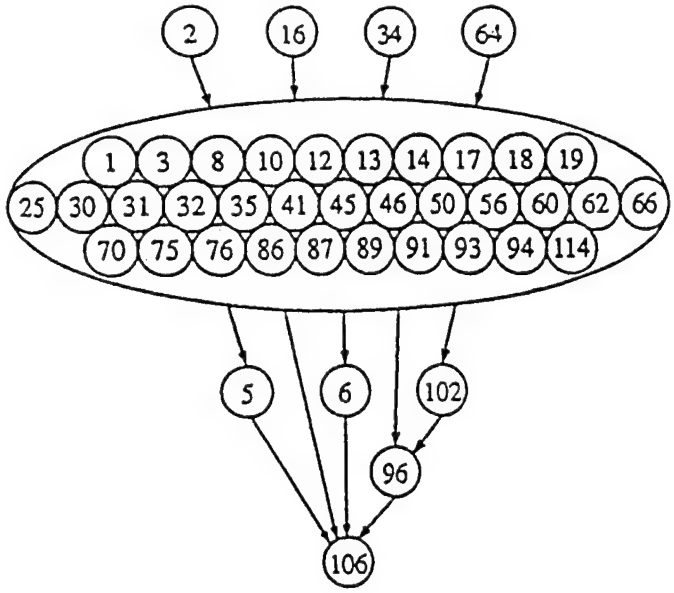


Figure 2: Composite Graph from Intersection of 21 Styles.

progress). Deadlock is the state in which a collection of processors cannot progress because they are cyclically waiting for input from one another. *Starvation* is the state in which some processors cannot progress because they are waiting for input from other processors (although the latter could be progressing normally). Deadlock arises if cycles are present in the flow graphs of the processors sending messages or if virtual "waiting" cycles may form due to finite buffers. Starvation is possible even if there are no cycles and buffers are infinite.

There are a number of approaches to solve the synchronization problems, many of which are described in an early survey by Misra (Misra 1986). In case of deadlock, Chandy and Misra (Chandy and Misra 1981) and Chandy, *et al.* (Chandy *et al.* 1983) suggest deadlock detection approaches, which let deadlock occur, detect it and then break it using collective global information. The former uses a central controller whereas the latter sends out queries for deadlock detection. However, several studies (Fujimoto 1988, Reed *et al.* 1988, Reed and Malony 88) indicate limited speed-up. These approaches do not address the starvation case.

The Null Message approach was developed independently by Bryant (Bryant 1977), Chandy-Misra (Chandy and Misra 1979a; Chandy and Misra 1979b) and Peacock *et al.* (Peacock *et al.* 1979a, Peacock *et al* 1979b). Messages which contain the current simulation time are sent from each processor to its successors so that the successors may proceed. However, the *look-ahead* provided by these null messages is limited (Peacock *et al.* 1979b, Nicol 1988) and studies show that choking of the simulation due to excessive null messages is likely (Fujimoto 1988, Reed *et al.* 1988, Reed and Maloney 88). (However, the studies were done on shared memory machines so the application of their results to distributed machines is subject to question.)

Peacock, *et al.* (Peacock *et al.* 1979a) also suggest a Blocking Table approach in which a processor blocks when the time of any of its predecessors is less than its own. However this approach involves one-to-many broadcasts to keep the tables current. Bezivin and Imbert (Bezivin and Imbert 1982) proposed a monitor approach and Christopher *et al.* (Christopher *et al.* 1982) a transaction based approach. Both involve centralized controllers which can be bottlenecks in a distributed system.

In the Appointment approach (Nicol and Reynolds 1984), processors demand appointment times from their predecessors and cannot simulate beyond the smallest of these times. The demand-driven Appointment approach has the advantage that it avoids unnecessary messages. Nicol (Nicol 1988) develops appointments further and provides greater look-ahead by doing application-dependent calculations. In case of stochastic networks, the service time is sampled probabilistically and it can be sampled even before the job arrives at the proces-

sor. Thus one can have a *future list* of events and higher appointment bounds, but its applicability is limited.

THE METHOD USED: APPOINTMENTS

In the near-term simulation described in this paper cycles between processors cannot be avoided. Starvation is also a concern since the output metrics cannot be obtained speedily if starvation occurs. Deadlock detection approaches to solving synchronization problems were deemed infeasible since they cannot prevent starvation. Null Messages were ruled out because of their limited look-ahead and probable choking. Monitor and transaction approaches were ruled out because of the possibility of bottlenecks.

Nicol and Reynolds' appointment approach seemed most promising and was adapted and enhanced for the near-term simulation. When a processor cannot make progress because of the unavailability of an event message from a predecessor, i sends a request message to that processor. The predecessor processor responds with an appointment time. Based upon the received appointment time the impeded processor may be able to process some buffered events and make progress.

The appointment time sent is not merely the current simulated time but rather is a time obtained by examining the buffers and the bundles in the processor. This application-dependent appointment time (similar to Nicol (Nicol 1988)) provides a good look-ahead.

When a request is sent by processor P_j to a predecessor processor P_i the next event time, t, to which P_j desires to advance is sent in the request. P_i, thus, only has to check its buffers for events which will complete before t. This saves P_i from having to check its entire buffer list. If an appropriate event destined for P_j is found by P_i then the time of that event is sent as an appointment. If there is no event which can complete before t, then P_i replies with a "you can proceed" message. If P_i has no pending event destined for P_j it sends a "failed" message.

P_j sends requests to those of its predecessors from whom it has no buffered messages. If the received messages are "you can proceed" messages or appointments, P_j can advance its clock to the smallest received appointment. If all the received messages are "you can proceed" then P_j can advance its clock to t. Upon receiving any "failed" messages, P_j waits for a small amount of time and restarts sending requests.

With this approach both deadlock and starvation are prevented. In terms of the design characteristics outlined by Reynolds (Reynolds 1988), this method is accurate, non-aggressive, has no risk and employs knowledge acquisition and knowledge embedding. It is non-aggressive because events are always processed in increasing time order and not on conditional knowledge. As a result the method is accurate; events *are* ultimately processed in increasing time order. There is no risk; events based on conditional knowledge are not propagated because there are no such events. Knowledge acquisition is used since the processors initiate requests for knowledge from other processors. Knowledge embedding is utilized because knowledge about the applications behavioral attributes is embedded in the simulation.

CONCLUSION

The complexity and performance requirements of the near-term simulator strongly suggest a parallel simulation. Unlike a single processor simulation, a parallel simulation gives rise to synchronization problems. These problems have been analyzed and an approach which enables an accurate, conservative simulation has been presented. The simulator has been completely implemented and real data obtained from an apparel plant is currently being processed. A pilot installation in a production apparel plant is expected to begin in summer 1991.

Future research will investigate algorithms for dynamically balancing the load across the processors of the transputer system. This balancing will be achieved by remapping operation buffers to processors with expected improvement in performance.

REFERENCES

Bezivin, J. and H. Imbert. 1982. "Adapting a Simulation Language to a Distributed Environment". In *Proceedings of the 3rd International Conference on Distributed Computing Systems*, (Ft. Lauderdale, FL), IEEE, N.Y., 596-603.

Bryant, R.E. 1977. "Simulation of Packet Communication Architecture Systems". Technical Report. MIT/LCS/TR-188, MIT, Cambridge, MA, (Nov.).

Christopher, T.; M. Evens; R.R. Gargeya; and T. Leonhardt. 1982. "Structure of a Distributed Simulation System". In *Proceedings of the 3rd International Conference on Distributed Computing Systems* (Ft. Lauderdale, Fl.), IEEE, N.Y., 584-589.

Chandy, K.M.; L.M. Haas; and J. Misra. 1983. "Distributed Deadlock Detection", *ACM Transactions on Computer Systems* 1, no. 2 (May), 144-156.

Chandy, K.M. and J. Misra. 1979a. "Distributed Sim-

ulation: A Case Study in Design and Verification of Distributed Programs", *IEEE Transactions on Software Engineering* SE-5, no. 5 (Sep.), 440-452.

Chandy, K.M. and J. Misra. 1979b. "Deadlock Absence Proofs for Networks of Communicating Processes", *Information Processing Letters* 9, no. 4 (Nov.), 185-189.

Foxfire Technologies Corporation. 1989. *Real-time Shop-Floor Control System User Manual*, Marietta, GA.

Fujimoto, R.M. 1988. "Performance Measurements of Distributed Simulation Strategies". In *Distributed Simulation 1988: Proceedings of the SCS Multiconference on Distributed Simulation* (Feb. 3-5), SCS, San Diego, CA, 14-20.

Jefferson, D.R. 1985. "Virtual Time", *ACM Transactions on Programming Languages and Systems* 7, no. 3 (Jul.), 404-425.

Khambekar, P.K. and S.K. Dharmaraj. 1990. "Approaches to Solving Synchronization Problems in Parallel Simulation of an Apparel Plant." In *Proceedings of the 1990 ACM Southeast Regional Conference* (Greenville, SC, April 18-20), ACM, N.Y., 274-281.

Misra J. 1986. "Distributed Discrete-Event Simulation", *ACM Computing Surveys* 18, no. 1 (Mar.), 39-65.

Nicol, D.M. and P.F. Reynolds. 1984. "Problem Oriented Protocol Design." In *Proceedings of the 1984 Winter Simulation Conference (16th)* (Nov. 28-30), 471-476.

Nicol, D.M. 1988. "Parallel Discrete-Event Simulation of FCFS Stochastic Queueing Networks." In *Proceedings of the ACM SIGPLAN PPEALS 1988* (Jul.), 124-137.

Pargas, R.P.; J.C. Peck; P.K. Khambekar; and S.K. Dharmaraj. 1990. "Near-term Distributed Simulation of Apparel Manufacturing." In *Proceedings of the 1990 Winter Simulation Conference* (New Orleans, LA, Dec. 9-12), SCS, San Diego, CA, 614-618.

Peck, J.C.; R.P. Pargas; P.K. Khambekar; and S.K. Dharmaraj. 1991. "Shop-Floor Performance Metrics for the Apparel Industry." Submitted to the *International Journal of Clothing Science and Technology.*

Peacock, J.K.; J.W. Wong; and E.G. Manning. 1979a. "Distributed Simulation Using a Network of Processors", *Computer Networks* 3, no. 1, 44-56.

Peacock, J.K.; J.W. Wong; and E.G. Manning. 1979b. "A Distributed Approach to Queueing Network Simulation." In *Proceedings of the 1979 Winter Simulation Conference* (San Diego, CA), IEEE, N.Y., 399-406.

Reed, D.A.; A.D. Malony; and B.D. McCredie. 1988. "Parallel Discrete Event Simulation Using Shared Memory", *IEEE Transactions on Software Engineering* SE-14, no. 4 (Apr.), 541-553.

Reed, D.A. and A.D. Malony. 1988. "Parallel Discrete Event Simulation: The Chandy-Misra Approach." In *Distributed Simulation 1988: Proceedings of the SCS Multiconference on Distributed Simulation* (Feb. 3-5), SCS, San Diego, CA, 8-13.

Reynolds, P.F. 1988. "A Spectrum of Options for Parallel Simulation." In *Proceedings of the 1988 Winter Simulation Conference*, (Dec.), 325-332.

Sokol, L.M., D.P. Briscoe; and A.P. Wieland. 1988. "MTW: A Strategy for Scheduling Discrete Simulation Events for Concurrent Execution." In *Distributed Simulation, 1988: Proceedings of the SCS Multiconference on Distributed Simulation* (Feb. 3-5), SCS, San Diego, CA, 34-42.

BIOGRAPHY

John C. Peck is a Professor of Computer Science at Clemson University. He received a B.S. in Mathematics, and M.S. and Ph.D. degrees in Computer Science, all from the University of Southwestern Louisiana. His research interests are in database systems and distributed algorithms. He is currently involved in developing support systems for manufacturing for the Defense Logistics Agency. He is also Vice-President for Research and Development of Foxfire Technologies, Inc. a company that designs real-time shop-floor control applications for the apparel industry.

Appendix E

GUIDELINES FOR DYNAMIC LOAD BALANCING IN CONSERVATIVE DISTRIBUTED SIMULATIONS

Roy P. Pargas and Prashant K. Khambekar
Department of Computer Science
Clemson University
Clemson, South Carolina 29634-1906

ABSTRACT

The problem of dynamically load balancing a distributed or parallel algorithm is difficult in and of itself. It becomes even more difficult when applied to a conservative distributed simulation primarily due to distributed logical clocks among the processors. Load cannot be arbitrarily transferred from one processor to another without first coordinating the respective clocks. The problem therefore is threefold: the distributed dynamic load balancing programs must monitor the processors for possible load imbalance, must collectively decide whether or not to balance load, and (if the decision is positive) must coordinate the clocks of the processors involved before transferring load. This paper addresses this issue and provides guidelines suggesting when and how dynamic load balancing of conservative distributed simulations may be carried out. Initial results of experimental runs are encouraging.

INTRODUCTION

In recent years, there has been growing interest in distributed simulations, i.e., simulations implemented on multiprocessors. This is partly due to the increased availability of relatively inexpensive multiprocessor systems, but mostly due to the need of increasingly sophisticated simulations for greater processing power (Chandy, Holmes and Misra 1979; Misra 1986; Nicol and Reynolds 1984; Pargas *et al.* 1990, Reed, Malony and McCredie 1988). It has been shown that dynamic load balancing can improve performance of parallel applications (Dragon and Gustafson 1989, Hinz 1990, Lin and Keller 1987, MaTM 1988, Saletore 1990, Shin and Chang 1989, Stankovic and Sidhu 1984, Willebeek-LeMair and Reeves 1989, Wu and Shu 1991) by moving load from a heavily loaded processor to a lightly loaded one. However, surprisingly little research has been conducted on applying dynamic load balancing techniques to distributed simulations. Because of temporal relationships among processors, the problem of load balancing a distributed simulation dynamically is much more difficult than load balancing other parallel algorithms. This paper attempts to provide guidelines suggesting when and how dynamic load balancing may be carried out.

A SMALL EXAMPLE

Consider the simulation example shown in Figure 1. The rectangles represent two processors, A and B (Figure 1a). Processor A has four job queues (Q1-Q4) and Processor B has one (Q5). Constant service times are given in parentheses. The arrows indicate the flow of jobs from one queue to the next. Each event is assumed to take the same amount of CPU time to process. Q1 and Q3 are assumed to have sufficient number of jobs in their queues at the start of the simulation and the other queues are empty.

Figure 1b gives a possible sequence of steps that processors A and B execute to process their job queues. The first column shows real-time units; the CPU requires one real-time unit to process each job. The columns "opn" refer to queues and the processing of jobs from each queue and the columns "lc" indicate the logical clocks. For example, the first job processed by A is a job from Q1 (opn=1) and completes at time lc=3. This is followed by a job from Q3 which completes at time lc=4. The third job is again from Q1 which completes at time 6. Two jobs complete at time 8, one from Q2 and another from Q3. The job from Q2 was the result of work passed to Q2 from Q1 at time lc=3.

Processor B works on only on jobs from Q5. The sequence of operations shows that processor B initially has to wait idly for a job from processor A and cannot perform useful computation until real time step 8. A total of 19 real time units is required for processors A and B to reach simulated times 23 and 24, respectively. If, however, Q4 is moved from processor A to processor B at real time 5 (Figure 1c), processor B is able to start performing useful computation sooner and processor A

reaches simulated time 23 after only 16 real time units. Moreover, processor B has gone far beyond simulated time 24. All in all, a significant improvement achieved through the movement of one job queue from A to B.

Though this example had an obviously unbalanced initial assignment of job queues to processors, the need for reassignment of load is observed even in applications where the initial assignment attempts to be equitable. Dynamic load balancing or load sharing may improve performance in such cases.

CLOCK CONSIDERATIONS

Moving a job queue from one processor to another requires moving jobs having very definite times associated with them. A queue may not be transferred unless the events in the queue are all scheduled in the receiving processor's future. Otherwise, the jobs will be processed out of time order violating a principal rule of conservative simulation. This problem is shown in Figure 2. Processor A's logical clock is at 12, past processor B's clock which is still at 10. This is possible because A's events are independent of B's events and need not wait for B's events to complete. If a queue is moved from B to A, it is possible that some messages received by processor C from processor A will be out of time order. In the example, a message from A with time stamp 12 arrives at C before a message with stamp 10. Note that the second message came from the queue that was moved from B to A. It is therefore necessary that if a queue is to be moved from one processor to another, the clock of the source processor should be greater than or equal to the clock of the destination processor. If the sending processor's clock is lagging, the receiving processor has to wait (do nothing, waste CPU time) for the sending processor to catch up and only then can a queue be transferred.

If we let $lc(A)$ and $lc(B)$ represent the logical clock values of processors A and B, respectively, and if B wants to transfer load to A, the role of the logical clock is summarized as follows.

if $lc(A) > lc(B)$
    B requests A to suspend operations
    B runs its clock forward to catch up with A's clock
endif
B transfers load to A

DYNAMIC LOAD BALANCING: THREE CASES

We define dynamic load balancing as the movement of load from one processor to another using information currently available to either or both of the processors. This definition therefore excludes the use of global information or communication among three or more processors in arriving at the load balancing decision.

Since only two processors are involved in any single load balancing decision, three cases are possible depending on job flow and the direction of job flow: (1) the processors are mutually independent, (2) one of the processors is dependent on the other, and (3) the processors are mutually dependent on each other.

Case 1. Processors are mutually independent.

If the processors are mutually independent and, say, A is leading (its logical clock is ahead of B's clock), A may potentially progress rapidly and complete the processing of its events long before B. In this situation, moving load from B to A may improve overall performance of the simulation.

How can A and B know that B's clock is lagging? Two possible ways are: (a) through periodic broadcast by both processors of their logical clock times, or (b) by having each processor pause and request the logical clock time of the other processor if a predefined period of time elapses without the processor having to suspend operation. Strategy (a) essentially informs each processor of the progress, or lack thereof, of all of the other processors. Strategy (b) assumes that if a processor is uninterrupted in its processing of events for an extended period of time, then it should check other processors' statuses for possible load imbalance.

In either case, after processors A and B determine that, say, B's clock is lagging, they must decide whether load balancing is appropriate. This decision may be based on factors such as the difference between their logical clock values and the number of queues in processor B (recall that movement of load implies movement of entire queues). If the decision is to perform load balancing, the clocks must be aligned as discussed in Section III above: A must suspend operation while B processes events and runs its clock forward until its clock either equals or exceeds A's clock; then B can transfer load to A.

Note that this transfer could change the dependence relationship between A and B, i.e., either or both processors may now be dependent on the other.

Case 2. B is dependent on A ($A \Longrightarrow B$). When job flow can occur from processor A to processor B then progress of B is dependent on messages from A. A is called a predecessor of B and B is called a successor of A. For an accurate conservative simulation, before a processor can simulate an event and advance its clock it should know that it will not get, from any of its predecessors, a message that needs to be processed at an earlier simulated

time. One of the better ways of performing this synchronization is by *appointments* (Nicol 1988). A processor demands appointments from its predecessors and cannot proceed beyond the smallest appointment received. Predecessor processors compute appointments by ***looking ahead*** based on their simulation state and predicting the smallest time before which they will not send a job message to the requesting successor. Good appointments enable a successor to proceed, possibly with multiple events, without any waiting, whereas bad ones cause it to wait idly till a suitable appointment or job message is received.

Table 1 lists the possible load balancing decisions for different situations when B is dependent on A. An asterisk (*) means "don't care" The first column asks if B's clock is lagging A's clock or vice versa. The second column, Number of Events, refers to the number of jobs that have been serviced. (As an aside, note that we assume that a job sitting in a queue may not be serviced because there is no agent to perform the service on the job. This is true in simulations of manufacturing plants, for example, in which jobs enter a queue but no worker is present at the workstation. The job remains in the queue. Jobs which *are* serviced constitute events and that is what we count in this column.) The third column asks if B is frequently waiting for jobs or appointments from A. The fourth column is the recommended load balancing decision.

The primary, although not exclusive, factor determining whether or not to balance load is whether A's or B's logical clock is lagging significantly behind the other processor's clock (the user defines what is and is not significant). If no clock lags, no load transfer is necessary. This is shown by the first line of Table 1 which indicates that no matter what else is occurring, if neither A nor B is lagging, no transfer is required.

If, on the other hand, A's or B's clock *does* lag, then one should consider other factors before deciding whether or not to transfer load. If A's clock lags, the lag can, generally speaking, be remedied by transferring load from A to B. However, if B is not waiting, then there should not be a transfer since B is busy, and A's bad appointments, if any, cannot be remedied as they are not caused by B. If B is waiting, and A processes either approximately equal or more number of events, then the decision is to transfer job queues from A to B since B is waiting. If B is waiting, and A processes less number of events, then a transfer is not warranted since B already more load.

If B's clock lags, the lag can be remedied by transferring load from B to A. If A and B process approximately equal numbers of events and B is waiting, the decision is to transfer load from B to A in the hope that after the transfer B will get better appointments from A. However, if B is not waiting, then B does not need better appointments and hence no transfer is warranted. If A processes less number of events, then the load should be transferred from B to A. If A processes more number of events and B is waiting then the load should be transferred from A to B to give B some work instead of idly waiting; however if B not waiting no load transfer should take place.

Case 3. A and B are mutually dependent on each other ($A \Longleftrightarrow B$).

When processors are mutually dependent on each other both give appointments to each other and thus may cause the other processor to wait depending on the quality of the appointments.

Table 2 lists the possible load balancing decisions for different situations. The first column asks if B's clock is lagging A's clock or vice versa. The second column, Number of Events, refers to the number of jobs that have been serviced. The third column asks if A is frequently waiting for jobs or appointments from B and the fourth column asks if B is, similarly, frequently waiting for jobs or appointments from A. The fifth column is the recommended load balancing decision.

Again the primary factor determining whether or not to balance load is whether A's or B's logical clock is lagging significantly behind the other's clock. If no clock lags, no load transfer is necessary. If A's clock lags and if either A and B process equal number of events or A processes more number of events than B, then the decision is to transfer from A to B in the hope of remedying the clock lag and also remedying bad appointments, if any. If A processes less number of events then the decision depends on whether A is waiting: if A is waiting load can be transferred from B to A otherwise the decision is not to transfer. The situation is symmetric if B's clock lags: if A processes equal or less number of events than B then the decision is to transfer from B to A, otherwise the decision depends on whether B is waiting.

RESULTS AND CONCLUSIONS

Dynamically load balancing a distributed simulation is significantly more difficult than load balancing most other distributed or parallel algorithms. This is because of the temporal relationships among the processors. The decision to transfer load from one processor to another is based on the differences in the logical clock values of each processor as well as the amount of load with each processor. It is not sufficient to base this decision only on a processor's current load but on its load

over some time interval. If a decision to balance load is made, more overhead is required, this time in the form of coordination of the logical clocks of the processors involved. The processor sending load must have a logical clock value greater than or equal to the logical clock of the receiving processor before the load is transferred.

Despite the significant overhead in dynamically load balancing a conservative distributed simulation, initial experimental results obtained with a manufacturing simulation have been encouraging. In six experiments involving 24 queues, 500 jobs, and 2 to 4 processors, dynamic load balancing algorithms produced an average of 8% percent improvement in CPU time. A detailed and complete tabulation of the full suite of the results of the experiments is currently being conducted.

REFERENCES

Chandy, K. M., V. Holmes and J. Misra. 1979. "Distributed Simulation of Networks". *Computer Networks* 3, no. 1 : 105-113.

Dragon, K. M. and J. L. Gustafson. 1989. "A Low-Cost Hypercube Load-Balance Algorithm". In *Proceedings of the Fourth Conference on Hypercubes, Concurrent Computers, and Applications* (Monterey, CA, Mar.) 583-589.

Hinz, D. Y. 1990. "A Run-time Load Balancing Strategy for Highly Parallel Systems". In *Proceedings of the Fifth Distributed Memory Computing Conference* (Charleston, SC, Apr.) 951-961.

Lin, F. C. H. and R. M. Keller. 1987. "The Gradient Model Load Balancing Method". *IEEE Transactions on Software Engineering* SE-13, no. 1 (Jan.): 32-38.

Ma, R. P., F. Tsung and M. Ma "A Dynamic Load Balancer for a Parallel Branch-and-Bound Algorithm". In *Proceedings of the Third Conference on Hypercube Concurrent Computers and Applications* (Pasadena, CA, Jan.) 1505-1513.

Misra, J. 1986. "Distributed Discrete-Event Simulation". *ACM Computing Surveys* 18, no. 1, (Mar.): 39-65.

Nicol, D. M. and P. F. Reynolds. 1984. "Problem Oriented Protocol Design". In *Proceedings of the 1984 Winter Simulation Conference (16th)* (Dallas, Texas, Nov. 28-30) 471-476.

Nicol, D. M. 1988. "Parallel Discrete-Event Simulation of FCFS Stochastic Queueing Networks". In *Proceedings of the ACM SIGPLAN PPEALS (Parallel Processing: Experience with Applications, Languages and Systems)* (Jul.) 124-137.

Pargas, R. P., J. C. Peck, P. K. Khambekar and S. K. Dharmaraj. 1990. "Near-term Distributed Simulation of Apparel Manufacturing". In *Proceedings of the 1990 Winter Simulation Conference* (New Orleans, LA, Dec 9-12) 614-618.

Reed, D. A., A. D. Malony and B. D. McCredie. 1988. "Parallel Discrete Event Simulation Using Shared Memory". *IEEE Transactions on Software Engineering* SE-14, no. 4, (Apr.): 541-553.

Saletore, V. A. 1990. "A Distributed and Adaptive Dynamic Load Balancing Scheme for Parallel Processing of Medium-Grain Tasks". In *Proceedings of the Fifth Distributed Memory Computing Conference* (Charleston, SC, Apr.) 994-999.

Shin, K.G. and Y. Chang. 1989. "Load Sharing in Hypercube Multicomputers for Real-time Applications". In *Proceedings of the Fourth Conference on Hypercubes, Concurrent Computers, and Applications* (Monterey, CA, Mar.) 617-621.

Stankovic, J. A. and I. S. Sidhu. 1984. "An Adaptive Bidding Algorithm for Processes, Clusters and Distributed Groups", In *Proceedings of the Fourth International Conference on Distributed Computing Systems* 4, (San Francisco, CA, May) 49-59.

Willebeek-LeMair, M. and A. P. Reeves. 1989. "Distributed Dynamic Load Balancing". In *Proceedings of the Fourth Conference on Hypercubes, Concurrent Computers, and Applications* (Monterey, CA, Mar.) 609-612.

Wu, M. and W. Shu. 1991. "Scatter Scheduling for Problems with Unpredictable Structures". In *Proceedings of the Sixth Distributed Memory Computing Conference* (Portland, OR, Apr.) 137-143.